Your Best Guess When You Know Nothing: Identification and Mitigation of Selection Bias

Katharina Dost✉, Katerina Taskova, Patricia Riddle, and Jörg Wicker
School of Computer Science
The University of Auckland
Auckland, New Zealand
✉ kdos481@aucklanduni.ac.nz

***Abstract*—Machine Learning typically assumes that training and test set are independently drawn from the same distribution, but this assumption is often violated in practice which creates a bias. Many attempts to identify and mitigate this bias have been proposed, but they usually rely on ground-truth information. But what if the researcher is not even aware of the bias?**

In contrast to prior work, this paper introduces a new method, IMITATE, to identify and mitigate Selection Bias in the case that we may not know if (and where) a bias is present, and hence no ground-truth information is available.

IMITATE investigates the dataset's probability density, then adds generated points in order to smooth out the density and have it resemble a Gaussian, the most common density occurring in real-world applications. If the artificial points focus on certain areas and are not widespread, this could indicate a Selection Bias where these areas are underrepresented in the sample.

We demonstrate the effectiveness of the proposed method in both, synthetic and real-world datasets. We also point out limitations and future research directions.

I. INTRODUCTION

Machine Learning typically assumes that training and test set are independently drawn from the same distribution (*i.i.d. assumption*), but this assumption is often violated in practice. For example, in the field of direct marketing, selected customers receive offers and their reactions form the data basis of the next prediction. Since they have not been drawn randomly but based on the highest expected outcome, this sample forms a biased subset and is not representative of all customers, although it is used to predict the behavior of all customers in order to decide over the next campaign [1]. Without adaption towards the entire customer group, this might result in a biased model yielding weak predictions and an unsuccessful campaign.

In contrast to prior work, in this paper, we are trying to solve the problem of Selection Bias when (a) we may not know if we have a bias, and (b) we have no ground-truth information on our dataset. Although Selection Bias in datasets is long known, it recently gained attention especially in the context of Fairness in Machine Learning [2]. Manifold attempts [3] [4] [5] to identify and mitigate the bias have been proposed, but they usually rely on ground-truth information and build upon the assumption that the researcher is aware of the bias.

But what if the problem already lies in the data gathering process? Unexpected Selection Biases can occur during the data collection phase. As an example, consider a fisherperson that collects information about fish species and amounts present in a lake. She throws her fishing net day by day and lists the fishes she caught, hoping to find a good distribution of the species. Depending on the fishing net she is using, small fishes might be able to escape since it is too coarse whereas large fishes might be strong enough to set themselves free.

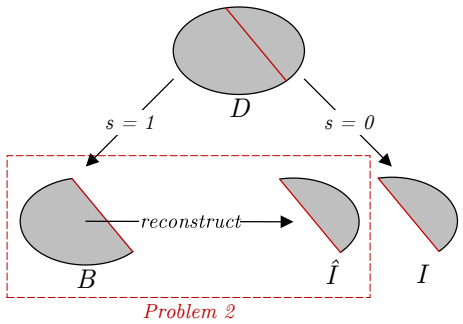

Fig. 1: **Problem 2:** A subset B of a dataset D is drawn according to a selection attribute s. The task is to reconstruct $I = D \setminus B$, resulting in a dataset $\hat{I}$.

Costly re-measurement or fragile domain adaption approaches can often be prevented if the bias is identified during the data collection process, e.g., the fisher could have replaced her net with a larger or smaller one if she had been aware of the bias she was creating.

Commonly used domain adaption approaches assume that the bias is known and an (unlabeled) sample of the ground-truth distribution or at least some information on it is available. That can be used to estimate the probability that a sample is present in the dataset, and re-sampling approaches [6] or sample weighting techniques [5] correct the model towards the ground-truth.

Whereas most Machine Learning techniques consider the data as given and mitigate the bias of the model instead of the dataset bias itself, we propose IMITATE (*Identify and Mitigate Selection Bias*), a technique that can be exploited in the early stage of data gathering and help the researchers improve the data quality by avoiding the bias in the first place.

IMITATE checks the data distribution and generates points to match a smooth probability density. If the artificial points

focus on certain areas, this could indicate a Selection Bias where these areas are underrepresented in the sample. The researcher can verify these areas by either using additional data from other sources or by extending her data gathering.

Although designed in a way that supports the data collection process, IMITATE is also applicable at a later stage as a preprocessing step that helps to prevent an induced bias in a model trained on the biased dataset.

Our main contributions are the following:

- We propose IMITATE, your best guess when you know nothing. Given only a (biased) dataset and no further information, IMITATE aims to imitate the ground-truth data in order to reveal a potential bias and mitigate it to enable the training of an unbiased model.
- Designing our method in a completely uninformed way is advantageous: IMITATE can be applied at a very early stage of the Data Mining process, i.e., during the data gathering phase when it is still possible to improve the data quality instead of accepting it as an immutable fact.
- Imitating the ground-truth improves the model trained on the dataset without interfering with the algorithm or the loss function to be optimized and hence constitutes a universely applicable preprocessing method that can be integrated into every Machine Learning pipeline.

The remainder of the paper is organized as follows: Sec. II provides the problem statement and the notation used throughout the paper. Sec. III gives an overview of related problems and their solutions, followed by Sec. IV that presents the proposed method, and Sec. V studies its behavior experimentally. Finally, Sec. VI concludes with discussion.

II. Problem Statement and Notation

We propose a method to solve the following problem statements that works in an unsupervised manner, i.e., it does not exploit any information about different classes. If it is applied in a supervised setting where several classes are present, the data is split according to the label and then the method is applied separately. Since the supervised setting offers possibilities for evaluation and comparison beyond domain expert consultation or interpretation, we formulate both problem statements here. See Fig. 1 for a visualization.

Problem 1 (Supervised Setting):
Let $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \subset \mathbb{R}^n \times L$ be an (unknown) n-dimensional real-valued labeled dataset consisting of m feature-label pairs $(\mathbf{x}_i, y_i)$. D is representative of an underlying distribution $\mathcal{D}$ that we consider as the *ground-truth*. A biased subset $B \subseteq D$ is drawn as follows: A selection variable s decides for each tuple $(\mathbf{x}_i, y_i) \in D$ if it is contained in B ($s = 1$), or discarded ($s = 0$). We consider s to be dependent on the present features or the class label which is typically known as MAR (*Missing At Random*) or MNAR (*Missing Not At Random*) bias [7]. Given only B, the goal is to approximate $I \coloneqq D \setminus B$ by a set $\hat{I}$ such that the gap between the original classification accuracy of a classifier trained on D and that of a classifier trained on $B \cup \hat{I}$ is minimal.

Problem 2 (Unsupervised Setting):
Assume the same setting as in the supervised case for a the one-class-case $|L| = 1$. Given only B, the goal is now to approximate $I \coloneqq D \setminus B$ as good as possible such that the distribution of $B \cup \hat{I}$ reflects $\mathcal{D}$.

As described above, a solution for Problem 2 can be extended to one for Problem 1 by treating every class label separately. If Problem 2 was solved and returned a good approximation of I, a similar classifier performance would be guaranteed. Note that the assumption that only B is available is a very strong restriction, but it enables us to detect *potential Selection Biases* of which researchers might not have been aware. The outcome needs to be carefully evaluated together with domain experts or validated using additional data from other sources.

III. Related Work

Although, to the best of our knowledge, there does not exist any literature on the exact same problem we are facing, several research fields deal with the problem under some additional assumptions. This section gives a rough overview of those assumptions and points to the particular research fields. Assume the problem setting introduced in the previous section.

If all the data points in D are available but only several features are missing for $x \in I$, we can use *Missing Value Imputation* to replace them and render I usable for training. The techniques range from basic approaches like imputing the average to more sophisticated methods like multiple imputation (see van Buuren and Groothuis-Oudshoorn [8] for an overview and example). Note that Missing Value Imputation does not necessarily require class labels.

Selection Bias refers to the case that all the data points in D are available (or at least their distribution) but only B is labeled and not I. Mehrabi *et al.* [2] provide a broad overview over different bias types and mitigation strategies in the context of Fairness in Machine Learning.

Slightly more general is the problem of *Covariate Shift* where the distribution of the test set differs from that of the training set without making any assumption on their relation to each other or the ground-truth. Both research fields, Selection Bias and Covariate Shift, stem from different backgrounds and focus on different applications but deal with similar problems: Given a labeled dataset consisting of data points X together with labels Y, they allow a shift in the data distribution $P[X|Y]$ between training and test set, but the label distribution $P[Y|X]$ is assumed constant and hence they aim to learn a classifier predicting $P[Y|X]$. Moreno-Torres *et al.* [3] present an overview of the related problems and different notations in this context whereas Rabanser *et al.* [4] review techniques to identify shifts. The most common practice in both fields is to use a weighted loss function during training that makes the resulting model more suitable for the test set [9] [10].

Different approaches to estimate the weights in different settings have been proposed, but all exploit the knowledge of the test set. Bickel *et al.* [5] give an overview over some of the

(a) Reconstructed Dataset

(b) Distribution Fitting per Dimension

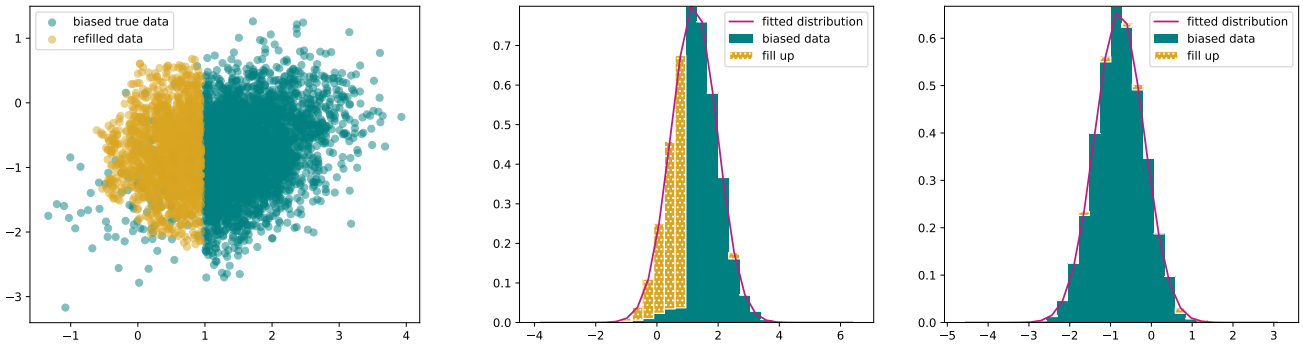

Fig. 2: The figure shows the central idea of IMITATE: (a) shows a dataset with a clear bias that has been "reconstructed". (b) shows both dimensions separately. The dataset is represented by a histogram, a normal distribution density is fitted to the histogram, and the gap between fitted and present density shows where to generate points.

approaches and introduce a method that models the selection variable and exploits it for the weighting. Smith and Elkan [7] train a generative classifier on arbitrarily biased training data, adapt it roughly to the test set using a weighting technique for bias and then gradually shift it towards a better test set representation exploiting the EM algorithm. Alberge *et al.* [11] mainly focus on image-classification tasks with neural networks and propose to compare the cumulative probability function resulting from a black-box predictor with that of the test set to detect a covariate shift. Stojanov *et al.* [12] investigate the influence of many dimensions on the common practices of covariate shift correction. A recent application of covariate shift correction techniques is image re-identification where a person needs to be spotted on images taken with different cameras in different environments and angles [13].

In the case that a concrete bias is suspected, different approaches to validate, quantify, and correct that bias have been proposed in different stages of the modeling process. Bellamy *et al.* [14] give an overview and an implementation in the *AI Fairness 360* toolkit. Focusing on Fairness in Machine Learning, they deal with a bias that advantages privileged groups and disadvantages others and mitigate it as a preprocessing step to learn a classifier, e.g., by reweighing the training set [15] or transforming the features [16].

If only the biased dataset B is known and no information on D is available, but the researcher suspects a certain bias, she might be able to find different data sources to either validate the underlying distribution of B or enrich it to be closer to the distribution of D. For example, consider a survey on baby care products that is sent out country-wide to all customers of a supermarket that are registered for a loyalty program. The responses will most likely not reflect the age and gender distribution of all customers or the entire population of the country. However, both can be verified using customer data or census data respectively.

If none of the above mentioned cases applies, we can only rely on the biased training set B and find ourselves in Problem 1. Bareinboim *et al.* [17] prove theoretically that in this case, the true label distribution $P[Y|X]$ cannot be recovered from the training data without additional external data or assumptions, which is a valid motivation for the heuristic approach we propose in this paper.

IV. PROPOSED METHOD

The problems introduced in Section II are important but hard problems and cannot be solved in all cases, but we will show in this section that in many cases there is something we can do. For example, a dataset measures the occurrences of different flower types together with geospatial coordinates, but for a certain area there are no measurements. In the case that the measurements are missing because the bespoken area is restricted and not publicly accessible, solving Problem 2 makes sense and might help overcome the bias. The method we propose in this section is able to imitate the measurements in this area. But if the data in this zone is missing because it is a lake and there simply are no flowers, an attempt to "reconstruct" the area would conceal the true distribution. The only way to distinguish between these two cases is to consult domain experts.

Aiming to solve the problems wherever useful, we introduce IMITATE[1] (*Identify and Mitigate Selection Bias*), a simple, modular, and extendable approach.

IMITATE mainly checks for underrepresented parts in the distributions to identify missing zones and monitors the confidence in its own results to decide if the output should be reported or discarded since it probably reflects noise or algorithm-inheritent problems. Figure 2 shows an example of the central idea. Given the biased dataset (turquoise in Fig. 2a), IMITATE measures the density for each (transformed) variable separately, e.g., in form of a histogram (Fig. 2b, turquoise bars). It then fits a Gaussian density function to the histogram

[1]For the reviewing period, we made our Python with Jupyter and Scipy [18] implementation (including all experiments and datasets) available under dropbox.com/sh/kyj9weohu20d2jl/AACilEXQ7ZXkDvN84V_rGveWa. Upon acceptance, we will move it to GitHub.

bins (Fig. 2b, pink lines) and fills up the gap between present and fitted distribution with generated datapoints (golden in both Fig. 2a and 2b).

Alg. 1 gives an overview over the main components of the algorithm. IMITATE takes a (biased) labeled dataset as input and separates the classes (Line 4). The subset X' is then transformed to another coordinate system where underrepresented parts of the distribution might be more clearly visible. For each dimension separately, the data density is represented by, e.g., a histogram which we use to fit a distribution density (Line 10). The gap between this estimated (unbiased) distribution $\hat{\mathcal{D}}$ and the present distribution is the area that we identify as $\hat{I}$. We then generate random data points in this area such that B including these points is distributed according to that estimated distribution $\hat{\mathcal{D}}$ (Line 14). The algorithm then estimates its confidence in the produced set of points and decides whether to keep or discard them (Line 21).

Note that we can use either domain expert knowledge or the confidence measure for all parameter choices that have to be made in IMITATE. If we use confidence, we cannot guarantee that the algorithm outputs the best possible solution, but the one in which it is most confident. The remainder of this section discusses the different elements of Alg. 1 in detail.

Algorithm 1 IMITATE: Simplified main algorithm

Input: A (biased) labeled dataset $B = (X, y) \subset \mathbb{R}^n \times L$
Output: A set of added labeled datapoints $\hat{I} = (\hat{X}, \hat{y})$

1: **function** IMITATE(X, y)
2: $\hat{X}, \hat{y} \leftarrow [\,]$
3: **for all** classes $c \in L$ **do**
4: $X' \leftarrow \{x_i \in X \mid y_i = c\}$ ▷ split classes
5: ▷ Remove outliers, transform coordinate system
6: $X' \leftarrow$ TRANSFORM(X')
7: **for all** dimensions $d \in \{1, \ldots, n\}$ **do**
8: ▷ Represent density over a grid, e.g., by KDE
9: $R_d \leftarrow$ REPRESENTDENSITY(X'_d)
10: $F_d \leftarrow$ FITDENSITY(R_d) ▷ Fit a Gaussian
11: $G_d \leftarrow F_d - R_d$ ▷ gap fitted vs. true data
12: **end for**
13: ▷ Generate points according to the gap distribution
14: $\hat{X}'_c \leftarrow$ FILLGAP($G_1, \ldots, G_n$)
15: ▷ Transform back to the original coordinate system
16: $\hat{X}_c \leftarrow$ TRANSFORMBACK($\hat{X}'_c$)
17: ▷ Estimate overall confidence in the result,
18: ▷ remove points with low individual confidence
19: $\hat{X}_c$, conf$_c \leftarrow$ REMOVELOWCONFIDENCE($\hat{X}_c$)
20: ▷ Store remaining generated points
21: $\hat{X}$.APPEND($\hat{X}_c$)
22: $\hat{y}$.APPEND($[c]$ * $|\hat{X}_c|$) ▷ add c until $|\hat{y}| = |\hat{X}|$
23: **end for**
24: conf $\leftarrow$ [conf$_1, \ldots,$ conf$_{|L|}$]
25: **return** $\hat{X}$, $\hat{y}$, conf
26: **end function**

A. Transformation

Given the input data belonging to one class, the feature space might need to be transformed into another space that gives a better view of the present probability densities. In order to do so, we first use the Local Outlier Factor technique [19] to remove outliers as a preprocessing step. This technique is beneficial since it is neighborhood-based and hence supports arbitrary dataset shapes (e.g., banana-like shapes) and does not require parameter choices.

In the examples provided in Fig. 2 there is a bias based on a selection variable that only relies on one of the features. This is easy to detect and does not need to be transformed in advance. However, if the selection variable depends on several features, bias in the densities over the axes will be less visible. Hence, we transform the data in another coordinate system in which the densities are aligned in a way that reveals missing areas (Fig. 2, gold).

Independent Component Analysis (ICA) [21] aims to reconstruct individual signals if only a weighted sum of them is known. It searches for a transformation to a new space described by independent components separating the individual signals. According to the *Central Limit Theorem*[2], those components that show the least similarities of the data density to a Gaussian are most likely individual signals whereas the Gaussian-like components indicate a mix of signals. Based on that observation, ICA searches for the independent components that show the least Gaussian density for a dataset.

Since we are interested in the components with the most visible deviation from the normal distribution, we use ICA as a heuristic and transform the dataset into the space defined by the components. Note that if all dimensions are normally distributed, ICA will not be able to find a solution and hence it is more likely that we improve the alignment using ICA than that we artificially generate a bias in the distribution that was not present beforehand.

B. Density Representations

The previous step outputs the data transformed to a new coordinate system. If we want to search for missing areas in the data distribution, we first need to find a discrete representation of the density that we have in the dataset, so we focus on one coordinate axis at a time. The simplest choice is a histogram. Histograms work well in many cases because they show a very clear drop of neighboring bin heights if there is a clear border between the biased and the present zone (see Fig. 2b left). However, unless the dataset is very large, they are sensitive to the choice of bin sizes and positions in the sense that the resulting histogram changes a lot if these parameters are slightly altered.

For smaller datasets (or large datasets with classes that contain only small amounts of examples) IMITATE uses *Kernel Density Estimators (KDE)* with Gaussian kernels to

[2]The *Central Limit Theorem* [20], an essential theorem from the mathematical field of probability theory, states that (under certain circumstances and proper normalization) the sum of independent random variables converges to a Gaussian.

represent the density. The estimator is then evaluated over a 1-dimensional equidistant grid which results in a similar representation to the histogram bins, but it is smoother.

We let the user choose the density representation. As a rule of thumb, we experienced in our experiments that classes of up to 5000 instances are well represented by a KDE whereas for larger sets the difference diminishes and the histogram representation is much faster. Either way, this step returns a 1-dimensional grid together with evaluations that are considered representative for the entire grid cell. Note that the granularity of the grid, i.e., the number of bins / grid cells, can be considered a user-defined constant or be chosen according to the highest confidence value.

C. Distribution Fitting

Based on a discrete representation of the data probability density, it should be possible to identify locations where data might be missing due to a bias. Therefore, we fit a distribution to that representation such that the density reveals these locations.

Many observations in real-life applications can be at least approximated by a normal distribution [22] since we can assume that several independent factors contribute to the output, see the Central Limit Theorem (Sec. IV-A). This is the main inspiration for IMITATE. Assuming that the original dataset D is normally distributed, the algorithm tries to fit a normal probability density function to the observed dataset B.

Given the density representatives $r_1, \ldots, r_{\#\text{bins}}$ of the dataset over a 1-dimensional grid with cell centers $g_1, \ldots, g_{\#\text{bins}}$ (the output of the density representation step; Sec. IV-B), we are aiming to fit a Gaussian to the estimates. A Gaussian is generally defined as a function of the form

$$g(x) = ae^{\frac{(x-b)^2}{2c^2}}$$

for $a, b, c \in \mathbb{R}$ and $c \neq 0$. In the case of $a = 1/(\sigma\sqrt{2\pi})$, $b = \mu$ and $c^2 = \sigma^2$ the Gaussian equals the probability density of a normally distributed random variable with mean μ and variance σ^2.

IMITATE uses the position of the highest bin as an initial estimate for the mean μ, calculates the variance σ^2 of the present data based on that mean, and adjusts the scaling factor a to match the highest bin value. Starting with these initial values, we use a weighted *Least Squares Optimizer* [23] to solve the optimization problem

$$\min_{a,b,c} \sum_{i=1}^{\#\text{bins}} w_i \left(g(g_i; a, b, c) - r_i\right)^2$$

with the weights $w_i = r_i^2$.

The weights are designed in a way that the optimizer puts more emphasis on high bins and is granted more freedom on the areas where very few or no data is present, i.e., the areas that we suspect the data is missing due to the bias. We omit further details on the choice of the weights here but provide them together with our implementation[3]. Looking at the example presented in Fig. 2b (left), equal weights would result in a probability density that is more shifted to the right in order to capture the bin heights below 1 correctly. Although that would be the closest fit to the present dataset, it does not indicate the potential biased regions. Once the optimal parameters for $\hat{a}, \hat{b}, \hat{c}$ are estimated, we evaluate $g(g_i; \hat{a}, \hat{b}, \hat{c})$ over the same grid and return the fitted values $r_1, \ldots, r_{\#\text{bins}}$.

In many cases, it is not possible to find one well fitted Gaussian, e.g., if the dataset consists of two clusters, IMITATE generates points between these clusters which is not what we aim for (we leave the problem of several clusters for Future Work; see Sec. VI). If the gap between the fitted and the present distribution gets very large, we assume that the result is not reliable anymore. To overcome this, we constraint $|\hat{I}| \leq \eta \cdot |B|$ for a user-defined η. In our experiments, we used $\eta = 1$ which seems to be a reasonable choice. If no solution can be found, this step returns $f_1, \ldots, f_{\#\text{bins}} = g_1, \ldots, g_{\#\text{bins}}$, the input values.

Algorithm 2 IMITATE: Fill in the Gap

Input: Gap vectors G_d and fitted density vectors F_d over 1D-grids with cell centers $g_1^d, \ldots, g_{\#\text{bins}}^d$ for each dimension $d \in \{1, \ldots, n\}$ restricted to one class c
Output: A set $\hat{X}_c'$ of generated data points for this class

```
function FILLGAP(G_1, ..., G_n)
    ▷ Determine the number of points to be generated
    #points ← max_d ||G_d||_1
    for all dimensions d ∈ {1, ..., n} do
        #points_d ← ||G_d||_1    ▷ Points for this dimension
        ▷ Convert to cumulative density function
        G'_d, F'_d ← CONVERTTOCDF(G_d, F_d)
        ▷ Get mixed cdf for this dimension
        p_d^G ← #points_d / #points
        cdf_d ← p_d^G · G'_d + (1 − p_d^G) · F'_d
        ▷ Draw #points coordinates according to cdf_d
        x_d ← RANDOM(cdf_d, #points)
    end for
    ▷ Combine single coordinate vectors and return
    return X̂'_c = [x_1^T, ..., x_d^T]
end function
```

D. Generation

Having transformed the data density into the representation $R = r_1, \ldots, r_{\#\text{bins}}$ and the fitted Gaussian evaluated over the same grid, $F = f_1, \ldots, f_{\#\text{bins}}$ (the outcomes of the steps described in Sec. IV-B and IV-C respectively), IMITATE next generates points that can lift the present density to the fitted one when added to the training set. These r_i and f_i have been calculated for each dimension and class separately and help decide in which grid cells data needs to be generated. Fig. 2b marks these cells in gold. The size of the golden bins

[3]We explored the effect of different weights and made it available together with the implementation as `Test_weights.ipynb`.

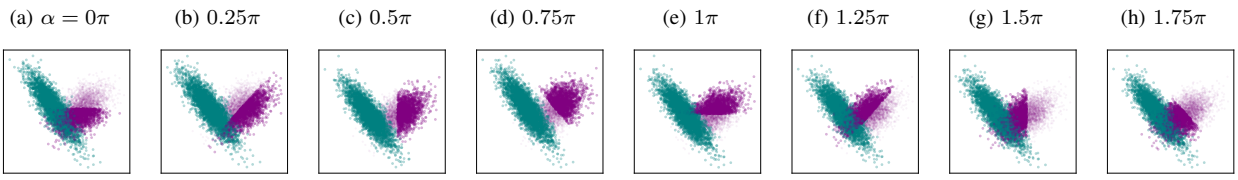

Fig. 3: We created synthetic biases on a dataset by rotating a horizontal cutting plane by an angle α. The points that are faded out describe I, the bold ones give B.

indicates the amount of points that are required to match the fitted distribution (pink line).

Algorithm 2 presents an overview of this step. For each dimension d and class c, we measure the gap between fitted and present density in order to determine how many points need to be added until this dimension's fitted density is achieved, i.e., $\#\text{points}_d = \sum_i \max\{f_i - r_i, 0\} = \|G_d\|_1$ where $\|\cdot\|_1$ denotes the $\ell 1$-norm (Line 3). Note that we use the maximum to focus only on the bins that need to be filled up. It allows us to ignore the case when $f_i < r_i$ for any i in which points from the input dataset would have to be removed in order to match the fitted density. The dimension with the highest gap determines how many points will be added in total, i.e., $\#\text{points} = \max_d \#\text{points}_d$.

Each dimension contributes its own coordinates to the final points. The grid cells are drawn in a way that the fitted density for this dimension is achieved. At first, the gap G_d is filled. If more coordinates need to be generated (i.e., because $\#\text{points}_d < \#\text{points}$), the remaining ones are drawn according to the fitted density F_d. To achieve that result, we obtain the mixed cumulative density function (*cdf*, Line 10) as

$$\text{cdf}_d = p_d^G \cdot \text{cdf}(G_d) + \left(1 - p_d^G\right) \cdot \text{cdf}(F_d)$$

for $p_d^G = \#\text{points}_d / \#\text{points}$ and use it to draw the cells in which the coordinates will then be drawn uniformly. That yields a vector x_d containing $\#$points coordinates for dimension d such that their addition to the original points B fulfills the fitted probability density over the grid (Line 15).

Once the coordinate generation is performed for each dimension, the results are combined into a set $\hat{X}'_c = \left[x_1^T, \ldots, x_d^T\right]$ of points for class c where $\cdot^T$ denotes the transposition of a vector. The result of this step can be seen in Fig. 2a: The golden points are the ones that IMITATE generated.

Note that the coordinate-wise generation of points is only able to model convex shapes. If the dataset has the shape of a ring (and IMITATE is not supposed to fill in the hole), another method needs to be found. This drawback will be addressed in future research (see Sec. VI).

E. Confidence

Having generated a set of points if we tested, we want to know if the results are good at all or should be discarded.

As the output of the algorithm, we expect clearly identified zones that are densely filled with generated points. If the points are spread over a wide area and rather singletons than clusters, that most likely only reflects noise in the data or a bad choice of the underlying grid (Sec. IV-B). We hence use a heuristic for the confidence of IMITATE in its output that compares the spread of the generated points to the spread of the dataset, i.e., for each point p we measure $d_{10\text{NN}}(p)$, the average distance to the 10 nearest neighbors. As a baseline for comparison, we use a random subset D' of the input dataset D with $|D'| = |\hat{I}|$, and average over the same calculated distance yielding $\mu_{10\text{NN}}(D')$ and the standard deviation $\sigma_{10\text{NN}}(D')$ from that mean. Other methods of comparison (such as the comparison to the spread of the class in the entire set D) have been tested. We omit them here due to space limitations and refer to our implementation[4].

All points p with $d_{10\text{NN}}(p) \geq \mu_{10\text{NN}}(D') + \sigma_{10\text{NN}}(D')$ are discarded right away. The confidence score is then defined as the average inverted 10-NN density

$$\text{conf} = 1 \Big/ \bar{d}_{10\text{NN}}(p).$$

A result will be discarded entirely if 10 or less points have been generated since the 10-NN-based distance measurement is not valid anymore, or if no points are left after the individual checks. If several parameter settings have been tested, e.g., grid granularities (#bins), we use the result with the highest confidence.

V. EXPERIMENTS

To carry out true real-world experiments we would need to find already biased datasets together with their ground-truth. But they are hard to find (which is why we need methods like IMITATE, after all!). Therefore, we study the behavior of the proposed method mainly on synthetic datasets where we can isolate the effects that we want to investigate, and then test IMITATE on real-world datasets with an artificially created bias. At the end, we give examples for how our technique should and should not be used.

A. Synthetic Data

For the experiments we generated synthetic 2D-datasets D with two classes (purple and turquoise) and, unless advised otherwise, 10000 samples and 5% label noise. The bias was created by a plane rotating around the center of one of the classes: p (default: $p = 0.05$) denotes the amount of datapoints above that plane that remained in B, the rest were removed.

[4]We tried different confidence measures and made our observations available together with the implementation as `Test_conf.ipynb`.

The other class and all points below the plane are contained in B. Fig. 3 shows the datasets with biases as described in the purple class where the planes were rotated. The rotation angles α are given in the captions. For the grid granularity, $\#\text{bins} \in \{5, 8, 11, \ldots, 29\}$ have been tested in each example and we use the result with the highest confidence. We repeat each experiment 10 times to make up for randomness and generate 10 datasets for each parameter setting.

We alter all three parameters (α, p, and the dataset noise) and report the improvement in accuracy in Figure 4. As a baseline, we train a linear SVM on the unbiased original dataset D and denote its accuracy on an unbiased test set as acc_D. Similarly, we train linear SVMs on B and on $B \cup \hat{I}$ and denote their performance on the same unbiased test set as acc_B and $\text{acc}_{B\cup\hat{I}}$, respectively. The dashed line shows the initial accuracy gap $\text{acc}_D - \text{acc}_B$, the solid line shows the final accuracy gap $\text{acc}_D - \text{acc}_{B\cup\hat{I}}$ after application of IMITATE. The smaller the value of the solid line, the better IMITATE performed the point generation.

Rotation α: As can be seen from the dashed line in Fig. 4a, the bias does not always affect the performance of the classifier. But if it does, IMITATE is able to find a set $\hat{I}$ that gives a substantial improvement of $B \cup \hat{I}$ over B alone. In the case of an $\alpha = 1.25\pi$ rotation bias in the purple class, we even see an improvement in the classifier performance if parts of the data are removed. The cases with high initial gaps are considered the interesting cases since there a bias mitigation is the most necessary. We use these cases for the following experiments.

Amount p of points in the biased area: Fig. 4b shows that for arbitrary amounts $p \in [0, 0.5]$ of points that remain in the biased area after application of the rotation bias, IMITATE can achieve a substantial improvement in performance. Note that if p is high enough, our technique to create a bias is not sufficient anymore to cause a biased classifier.

Label Noise in the dataset: The higher the label noise in the dataset, the less the synthetic bias affects the results. We can see that trend in Fig. 4c. It is also clearly visible that IMITATE helps more when there is only a small amount of noise present and can even decrease the performance of a classifier for noisy data. That is the expected result as too much noise overshadows the true probability density and leads to an incorrectly fitted density.

B. Real-World Data

For the evaluation of our method on real-world data, we used classification datasets from the *UCI Machine Learning Repository*[5] [24] and created a bias as follows: To make sure that our split is relevant for the classification, a decision stump was trained on each dataset and we used the same split for the bias and removed the corresponding attribute afterwards. B is then the data in the larger leaf, I is the smaller one. If that procedure would have removed a class (almost) completely, we used a decision stump on the data without the original attribute. Table I summarizes the datasets together with the criteria for the bias.

[5]Thanks to NASA for allowing us to use the shuttle datasets.

Dataset	**Predicted Attribute**	**Biased Set B**
Abalone	Sex	Viscera weight < 0.144
Banknote*	Class	Variance > 0.32
Car**	Class	Persons > 3
Statlog(Shuttle)	Class==Rad Flow	Time > 54.5
Skin* [25]	Class	R ≤ 170.5

TABLE I: Dataset Overview. *Due to the small dimensionality of the dataset, we did not omit the attribute used for the bias split. **The categorical variables were transformed to integers as a preprocessing.

Dataset	**Baseline Classifier**	**Initial Gap** $\text{acc}_D - \text{acc}_B$	**Final Gap** $\text{acc}_D - \text{acc}_{B\cup\hat{I}}$
Abalone	SVM (linear kernel)	0.151	0.089*
	SVM (RBF kernel)	0.151	0.107*
	Decision Tree	0.154	0.103*
Banknote	SVM (linear kernel)	0.218	0.176*
	SVM (RBF kernel)	0.205	0.162*
	Decision Tree	0.196	0.156*
Car	SVM (linear kernel)	0.133	0.144
	SVM (RBF kernel)	0.131	0.138
	Decision Tree	0.146	0.152
Shuttle	SVM (linear kernel)	0.607	0.466*
	SVM (RBF kernel)	0.590	0.424*
	Decision Tree	0.566	0.482
Skin	SVM (linear kernel)	0.003	-0.010*
	SVM (RBF kernel)	0.002	-0.010*
	Decision Tree	0.005	-0.009*

TABLE II: We show the initial accuracy gap between a baseline classifier trained on the ground-truth dataset D and one trained on the biased set B and compare it to the final gap (after application of IMITATE). * indicates that the result is statistically significantly better than the other (t-test at significance level of 1%).

We measured the performance of IMITATE by the same accuracy gaps as in Sec. V-A. Different baseline classifiers were used, i.e., SVMs with a linear and an RBF kernel as well as a Decision Tree. In order to obtain more reliable results, we repeated each experiment 10 times and reported the average result which is displayed in Table II. The * indicates that one result was significantly better than the other based on a t-test at significance level of 1%.

Discussion: For the Abalone, Banknote, Shuttle, and the Skin dataset, IMITATE could significantly improve the biased dataset in order to obtain better classification results. The Skin dataset allowed us to improve over the original result. Although that is a desirable result, it implies that IMITATE did not reconstruct the original data but generated new data. The fact that it could improve the performance indicates that there might very well be a bias on the original dataset! On the Car dataset, our technique does not improve the data quality. That is due to the fact that the Car dataset is discrete and hence very sensitive to the choice of the underlying grid (see Sec. IV-B). IMITATE mainly fills in the gaps between the true values (with a high confidence) instead of focusing on other underrepresented areas. Discrete datasets are a weakness of

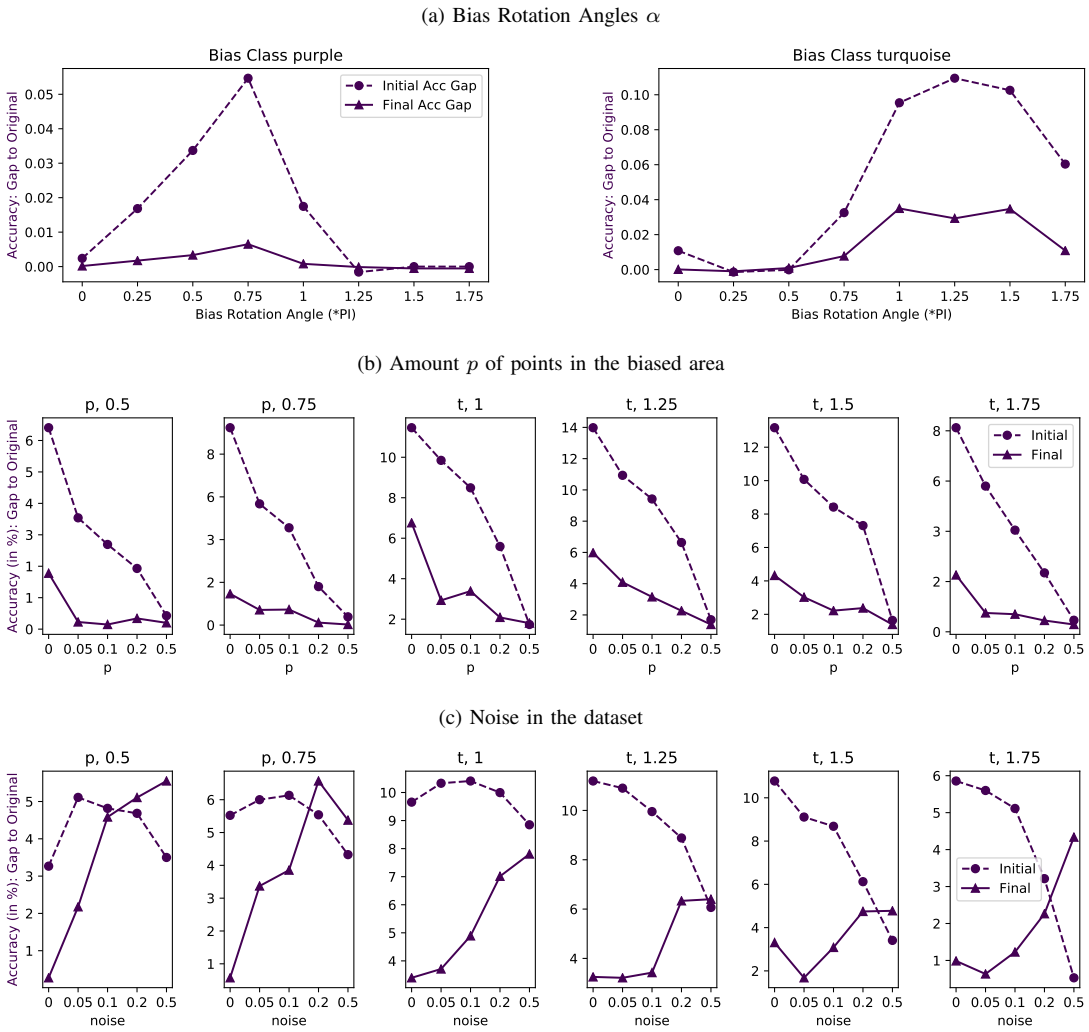

Fig. 4: Experiments on synthetic data using synthetic rotation biases. The plots (a), (b), (c) explore the behavior of IMITATE for different rotation angles (α), amounts of remaining points in the biased area (p), and noise in the dataset, respectively. Reported is the gap in accuracy between a classifier trained on the original unbiased dataset D and one that trained on B only (dashed line) and on $B \cup \hat{I}$ (solid line). If it hits 0, we managed to fully reconstruct the performance of the unbiased classifier. For (b) and (c), the type of synthetic bias is given by the title $(c, \alpha/\pi)$ which means a rotation bias in class $c \in \{\text{purple (p)}, \text{turquoise (t)}\}$ with angle α.

IMITATE that we need to overcome, e.g., by adding noise or dimensionality-reduction, as discussed in Sec. VI.

C. Use-Case: Cardiovascular Disease

In order to show how IMITATE can be applied in the data gathering process, we use Kaggle's Cardiovascular Disease dataset[6]. The dataset contains 70000 medical examination measurements together with factual information on the patient and the target variable states the presence or absence of cardiovascular disease. Since we want to be able to visualize it easily, we restrict the dataset to the age and the weight of a patient as well as the class label and apply IMITATE to it.

Figure 5 shows the result. The upper two plots show the probability densities and the areas in which points are supposed to be generated for the two classes respectively. The lower plot then gives the result: the original data B is faded out, the generated data points $\hat{I}$ are the bold ones.

The result clearly shows two trends: (i) There is a large amount of data missing for the patients over 65 years, especially for ones suffering from cardiovascular disease. It is well known that the risk of cardiovascular disease increases with the age of the patients, but the dataset shows a hard border here. Hence we consider this as a reasonable result. (ii) Especially for the healthy patients but also for the other ones we see a generated area below a weight of 60kg. This result could have several reasons that a domain expert should carefully assess, e.g., it might mirror the observation that there are more over- than underweight people in the dataset. That is possibly due to the fact that overweight increases the risk of cardiovascular

[6] kaggle.com/sulianova/cardiovascular-disease-dataset

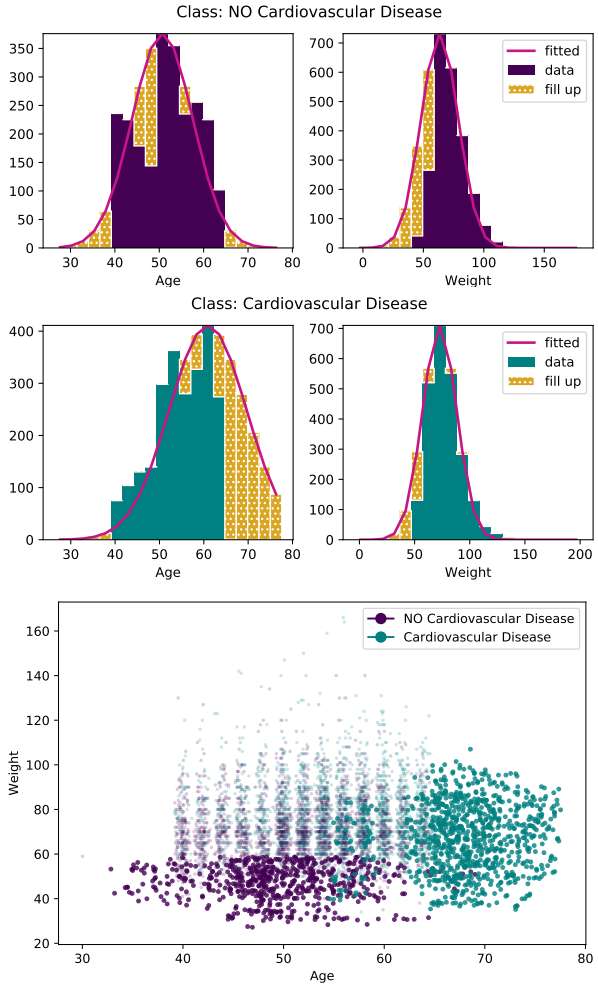

Fig. 5: We apply IMITATE to Kaggle's Cardiovascular Disease dataset, restricted to the age and weight features. The upper two plots show the data densities per axis for both classes (healthy and sick), respectively. The lower plot demonstrates the final result ($\hat{I}$ is bold, B is faded out).

disease which is why overweight people are more likely to go for a corresponding examination, but maybe the bias here also reflects a trend in the underlying weight distribution of the entire population in the area the data was collected.

If we want to validate these results, we could either consult a domain expert or use additional datasets to check the distributions of the underlying population (i.e., the potential patients of the location where the data was collected) in terms of weight or age and compare it to what we found. These underlying distributions can then be exploited for the typical weighting approaches or other covariate shift methods to train a classifier that is reliable for the entire population, not only for the patients of this specific location.

D. Drawbacks: Clusters and Hard Boundaries

So far, we saw that IMITATE performs well in many cases and helps us identify potential biases. However, we need to

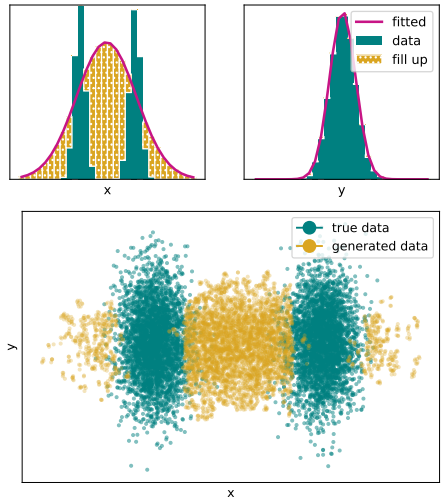

Fig. 6: Drawbacks of IMITATE: We applied our method to a synthetic dataset consisting of two clusters (turquoise). IMITATE fills in the gap between the clusters (gold).

be careful with either discrete datasets or datasets with several clusters per class for the same reason: IMITATE fills in the gap between the clusters as shown in Figure 6 and thereby overshadows potential biases inside the clusters.

Fig. 6 shows an extreme case. If the two clusters were closer together, the effect would be less harmful. If they were much further apart from each other, the restriction that we allow IMITATE to at most double the points (see Sec. IV-C) kicks in and results in no correction for this particular dimension at all. That means that biases in this dimension are not analyzed, but at least the result is not harmful.

A discrete dataset basically yields the same problem, but it can be mitigated by the right choice of the underlying grid as the discreteness can be smoothed out if the grid is coarse enough and well positioned.

Another drawback of IMITATE are hard domain-dependant boundaries, e.g., in the example explained at the beginning of Sec. IV where we wanted to "complete" a dataset of flower measurements in an area with a lake. The lake here is a hard boundary and no reconstruction over the lake area should be made. In another identical dataset, the gap could have occurred due to a restricted area where no measurements could be taken but there are flowers. We cannot expect IMITATE to automatically distinguish between those identical datasets, but we should allow for user-given hard boundaries in the density fitting process.

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduced IMITATE, a simple, modular, and extendable approach to identify and mitigate Selection Bias in the case that we may not know if (and where) we have a bias, and hence no ground-truth information is available. In contrast to comparable methods that consider the data as given and exploit background information to learn a back-shifted

classifier, IMITATE can be used in the data gathering process to identify a potential bias right away.

Experiments showed that IMITATE is able to yield meaningful results and can support the data collection process by pointing out potential biases in an early stage, but it is also capable of bias mitigation in a later stage of the Data Mining process. We discovered that fitting one Gaussian per (transformed) feature often helps, but does not always lead to success. The major problems that need to be addressed in future research are the following.

Discrete Dataset: If the dataset is discrete and not continuous, IMITATE's performance relies heavily on the choice of the underlying grid for the density estimation and fitting. An equi-distant grid is hard to adjust and might not always be a suitable choice. Extensions of IMITATE hence should include an automated way of finding a well-suited grid for each dimension individually. Other options to smooth out the discreteness here could be adding noise or dimensionality reduction.

Clusters: If the dataset consists of several clusters per class, in the dimensions that separate the clusters, IMITATE will either fill in the gap between the clusters or not do anything at all if the clusters are too far away from one another. The second case is fine as long as there are dimensions where the clusters overlap; the first case is a problem. Improvements to IMITATE should take that into account and either apply a pre-clustering and treat each cluster separately or fit a mixture model of several Gaussians depending on the number of clusters that show in a particular dimension.

Hard Boundary: We cannot determine if a dataset has a hard boundary somewhere (or if such a boundary is a sign for a bias) as boundaries are domain-related. However, we will extend IMITATE to allow users to set constraints that represent these boundaries and consider them during the density fitting. One way would be by adjusting the weights in the optimization accordingly. If the dataset is shaped like a ring with a circular boundary in the middle (e.g., the lake-problem in Sec. IV and V-D), a re-adjustment of the weights will not be sufficient. In this case, a solution might be to transform the dataset into a higher-dimensional space that is able to separate both areas by a plane.

Overall, IMITATE is the first method to identify and mitigate selection bias when no ground-truth or additional knowledge is required. We see in IMITATE a promising start of a new direction of research that is modular enough to allow for extensions and improvements.

REFERENCES

[1] S. N. Bhaduri and D. Fogarty, *Mitigating Sample Selection Bias Through Customer Relationship Management*, pp. 71–83. Singapore: Springer Singapore, 2016.

[2] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," 2019.

[3] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognition*, vol. 45, pp. 521–530, Jan. 2012.

[4] S. Rabanser, S. Günnemann, and Z. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," in *Advances in Neural Information Processing Systems 32*, pp. 1396–1408, Curran Associates, Inc., 2019.

[5] S. Bickel, M. Brückner, and T. Scheffer, "Discriminative learning for differing training and test distributions," in *Proceedings of the 24th International Conference on Machine Learning*, ICML 07, (New York, NY, USA), pp. 81–88, 2007.

[6] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML 04, (New York, NY, USA), p. 114, 2004.

[7] A. T. Smith and C. Elkan, "Making generative classifiers robust to selection bias," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 07, (New York, NY, USA), p. 657666, 2007.

[8] S. van Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by Chained Equations in R," *Journal of Statistical Software*, vol. 45, pp. 1–67, Dec 2011.

[9] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Scholkopf, "Correcting sample selection bias by unlabeled data," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS 06, (Cambridge, MA, USA), pp. 601–608, 2006.

[10] A. Liu and B. D. Ziebart, "Robust classification under sample selection bias," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS 14, (Cambridge, MA, USA), pp. 37–45, 2014.

[11] F. Alberge, C. Feutry, P. Duhamel, and P. Piantanida, "Detecting Covariate Shift with Black Box Predictors," *2019 26th International Conference on Telecommunications, ICT 2019*, pp. 324–329, 2019.

[12] P. Stojanov, M. Gong, J. Carbonell, and K. Zhang, "Low-dimensional density ratio estimation for covariate shift correction," *Proceedings of Machine Learning Research*, vol. 89, pp. 3449–3458, Apr 2019.

[13] L. Song, C. Wang, L. Zhang, B. Du, Q. Zhang, C. Huang, and X. Wang, "Unsupervised Domain Adaptive Re-Identification: Theory and Practice," *Pattern Recognition*, vol. 102, p. 107173, 2020.

[14] R. K. E. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, and K. Kannan et al., "Ai fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias," *IBM Journal of Research and Development*, vol. 63, no. 4/5, pp. 1–15, 2019.

[15] F. Kamiran and T. Calders, "Data Preprocessing Techniques for Classification Without Discrimination," *Knowledge and Information Systems*, vol. 33, pp. 1–33, Oct 2012.

[16] F. P. Calmon, D. Wei, B. Vinzamuri, K. N. Ramamurthy, and K. R. Varshney, "Optimized pre-processing for discrimination prevention," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS 17, (Red Hook, NY, USA), pp. 3995–4004, 2017.

[17] E. Bareinboim, J. Tian, and J. Pearl, "Recovering from selection bias in causal and statistical inference," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI 14, pp. 2410–2416, 2014.

[18] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, and Cournapeau et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[19] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD 00, (New York, NY, USA), pp. 93–104, Association for Computing Machinery, 2000.

[20] W. Hoeffding and H. Robbins, "The central limit theorem for dependent random variables," *Duke Mathematical Journal*, vol. 15, pp. 773–780, Sep 1948.

[21] A. Hyvärinen and E. Oja, "Independent Component Analysis: Algorithms and Applications," *Neural Networks*, vol. 13, pp. 411–430, Jun 2000.

[22] A. Lyon, "Why are Normal Distributions Normal?," *British Journal for the Philosophy of Science*, vol. 65, no. 3, pp. 621–649, 2014.

[23] J. J. Moré, "The Levenberg-Marquardt Algorithm: Implementation and Theory," in *Numerical Analysis* (G. A. Watson, ed.), pp. 105–116, Springer, Berlin, Heidelberg, 1978.

[24] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[25] R. Bhatt and A. Dhall, "Skin segmentation dataset." UCI Machine Learning Repository.