Fairness-Driven Private Collaborative Machine Learning

DANA PESSACH, Tel-Aviv University, Israel
TAMIR TASSA, The Open University, Israel
EREZ SHMUELI*, Tel-Aviv University, Israel

The performance of machine learning algorithms can be considerably improved when trained over larger datasets. In many domains, such as medicine and finance, larger datasets can be obtained if several parties, each having access to limited amounts of data, collaborate and share their data. However, such data sharing introduces significant privacy challenges. While multiple recent studies have investigated methods for private collaborative machine learning, the fairness of such collaborative algorithms was overlooked. In this work we suggest a feasible privacy-preserving pre-process mechanism for enhancing fairness of collaborative machine learning algorithms. An extensive evaluation of the proposed method shows that it is able to enhance fairness considerably with only a minor compromise in accuracy.

CCS Concepts: • **Security and privacy → Privacy-preserving protocols**; • **Computing methodologies → Artificial intelligence**; **Machine learning**; • **Information systems → Information systems applications**.

Additional Key Words and Phrases: Privacy, Algorithmic Fairness, Collaborative Machine Learning, Federated learning, Secure Multi-Party Computation

1 INTRODUCTION

Machine learning (ML) is one of the most prominent technological developments in recent years, rapidly becoming a central part of our life. Applications of ML are all around us, ranging from traffic prediction and virtual personal assistants to automated radiology and autonomous vehicles.

The performance of ML models inherently depends on the availability of a large quantity of useful training data. In neural network learning, for example, recent studies have shown that accuracy can be improved considerably by having access to large datasets [1]. In many applications, however, data is scattered and held by multiple different parties that may be reluctant to share their information for multiple reasons such as commercial competition, privacy concerns, and in some domains even legal constraints. For example, the Health Insurance Portability and Accountability Act (HIPAA) in the United States places strict constraints on the ability of health care providers to share patient data [2].

As a result, a variety of methods have been recently proposed to allow multiple parties to collaboratively train ML models while preventing the disclosure of private information. Hereinafter, we refer to this setting as *private collaborative machine learning* (PCML). While these methods address a very similar problem, they are often associated with different (though overlapping) research domains including *privacy-preserving data mining* [3, 4], *privacy-preserving machine learning* [5], *collaborative machine learning* [6, 7], and *federated machine learning* [8].

These methods take two main approaches for preserving privacy. The first approach is based on *perturbation* [4], which incorporates noise to the training data in order to obscure sensitive information. *Differential privacy* [9] is perhaps the most prominent perturbation technique. The main limitation of this approach is that incorporating noise to the data may yield an inferior model. The second approach is based on *secure multi-party computation* (MPC) [3]. MPC achieves

*Corresponding author

Authors' addresses: Dana Pessach, Tel-Aviv University, P.O. Box 39040, 6997801, Tel-Aviv, Israel, danapessach@gmail.com; Tamir Tassa, The Open University, P. O. Box 808, 4353701, Ra'anana, Israel, tamirta@openu.ac.il; Erez Shmueli, Tel-Aviv University, P.O. Box 39040, 6997801, Tel-Aviv, Israel, shmueli@tau.ac.il.

privacy protection by applying cryptographic techniques that enable several parties to perform a joint computation on private data that is distributed among them, where only the outcome of the computation is revealed to the parties, but no other information is exposed [10]. In contrast to perturbation, MPC does not change the data, and therefore the output issued by such algorithms is identical to the output of their non-privacy-preserving counterparts. However, since in many cases, a generic and perfectly secure MPC solution is infeasible [11], lower security requirements are typically accepted for the sake of higher efficiency [12].

Despite the growing number of studies proposing algorithms for PCML, the fairness of such algorithms was overlooked. Since many automated decisions (including which individuals will receive jobs, loans, medication, bail or parole) can significantly impact peoples' lives, there is great importance in assessing and improving the fairness of the decisions made by such algorithms. Indeed, in recent years, the concern for algorithmic fairness has made headlines. One of the most prominent examples was in the field of criminal justice, where recent revelations have shown that an algorithm used by the United States criminal justice system had falsely predicted future criminality among African-Americans at twice the rate as its parallel predictions for white people [13]. These lines of evidence and concerns about algorithmic fairness have led to growing interest in the literature on defining, evaluating and improving fairness in ML algorithms (see a recent review in [14, 15]). All of these studies, however, focused on centralized settings.

In this paper, we consider a learning setting similar to the PCML setting described above, where data is scattered among several parties, who wish to engage in a joint ML procedure, without disclosing their private information. Our setting, however, also adds a fairness requirement, mandating that the learnt model satisfies a certain level of fairness.

To address the new fairness requirement, we suggest a privacy-preserving pre-process mechanism for enhancing fairness of collaborative ML algorithms. Similarly to the pre-process fairness mechanism suggested in [16], our method improves fairness through decreasing distances between the distributions of attributes of the privileged and unprivileged groups. Our approach is not tailored to a specific algorithm, and therefore can be used with any PCML algorithm. In contrast to [16], our method was designed to allow privacy-preserving enhancements, which are obtained through MPC techniques.

An extensive evaluation that we conducted over real-world datasets shows that the proposed method is able to improve fairness considerably, with almost no compromise in accuracy. Furthermore, we show that the runtime of the proposed method is feasible, especially considering that it is executed once as a pre-process procedure.

The paper is organized as follows. We begin with a review of related work in Section 2. In Section 3 we define the problem that we study and present relevant notations that we shall be using. In Section 4 we describe our solution to this problem — a privacy-preserving fairness-enhancing mechanism. We present our experimental evaluation in Section 5, and conclude in Section 6.

2 RELATED WORK

We organize the survey of relevant related work as follows. In Section 2.1 we survey related studies that deal with algorithmic fairness in centralized settings. In Section 2.2 we review literature on private collaborative machine learning. We conclude, in Section 2.3, with a discussion of recent research efforts on private and fair machine learning. We note that all of the latter studies focused on either a centralized setting or a distributed setting which is non-collaborative, while we consider in this work a distributed collaborative setting.

2.1 Algorithmic Fairness in Centralized Settings

Fairness Definitions and Measures

The law makes a distinction between two types of discrimination: i) *Disparate treatment* [17, 18]: intentionally treating an individual differently based on his/her membership in a protected class/unprivileged group (*direct discrimination*); and ii) *Disparate impact* [18, 19]: negatively affecting members of a protected class/privileged group, more than others, even if by a seemingly neutral policy (*indirect discrimination*).

Put in our context, it is important to note that algorithms trained with data that do not include sensitive attributes (i.e., attributes that explicitly identify the privileged and unprivileged groups) are unlikely to produce *disparate treatment*, but may still induce unintentional discrimination in the form of *disparate impact* [20].

In the algorithmic fairness literature, multiple measures were suggested. (The reader is referred to [14, 15, 21] for a comprehensive review of fairness definitions and measures.) The most prominent measures include *demographic parity* [22, 23] and *equalized odds* [24]. *Demographic parity* ensures that the proportion of the positive predictions is similar across groups. For example, if a positive prediction represents acceptance for a job, then the demographic parity condition requires the proportion of accepted applicants to be similar across groups. One disadvantage of this measure is that a fully accurate classifier may be considered unfair, when the base rates (i.e., the proportion of actual positive outcomes) of the various groups are considerably different. Moreover, in order to satisfy *demographic parity*, two similar individuals may be treated differently since they belong to two different groups, and in some cases, such treatment is prohibited by law.

In this paper we focus on a variation of the *equalized odds* measure. This measure was devised by [24] to overcome the disadvantages of measures such as *demographic parity*. It was designed to assess the difference between the two groups by measuring the difference between the false positive rates (FPR) in the two groups, as well as the difference between the false negative rates (FNR) in the two groups.

In contrast to the *demographic parity* measure, a fully accurate classifier will necessarily satisfy the two *equalized odds* constraints. Nevertheless, since *equalized odds* relies on the actual ground truth, it assumes that the base rates of the two groups are representative and were not obtained in a biased manner.

It is important to note that there is an inherent trade-off between accuracy and fairness: as we pursue a higher degree of fairness, we may compromise accuracy (see, for example, [20]).

Mechanisms for Enhancing Fairness

Fairness-enhancing mechanisms are broadly categorized into three types: pre-process, in-process and post-process. Pre-process mechanisms involve changing the training data before feeding it into the ML algorithm [16, 25–29]. For example, [16] suggested to modify the attributes in the dataset, so that the distributions for both privileged and unprivileged groups become closer, and therefore making it more difficult for the algorithm to differentiate between the two groups. In-process mechanisms involve modifying ML algorithms to account for fairness during training time [22, 27, 29–39]. For example, [30] suggested adding a regularization term to the objective function that penalizes the mutual information between the sensitive attribute and the model's predictions. Post-process mechanisms perform post-processing to the output scores of the model to make decisions fairer [24, 40–42]. For example, [24] proposed a technique for flipping some decisions of the model in order to enhance equalized odds. Similarly, [40] suggested to select a threshold for each group separately, in a manner that maximizes accuracy and minimizes demographic parity.

The different mechanism types are associated with the following advantages and disadvantages [14, 15]. Pre-process mechanisms can be advantageous since they can be used with any ML algorithm. However, since they are not tailored to a specific algorithm, there is no guarantee on the level of accuracy that such a mechanism may yield at the end of the process. Similar to pre-process mechanisms, post-process mechanisms may be used with any ML algorithm. However, due to the relatively late stage in the learning process in which they are applied, post-process mechanisms typically obtain inferior results [31]. Another danger of post-process mechanisms is that they may treat entirely differently two individuals who are very similar across all attributes except for the group to which they belong. In-process mechanisms are beneficial since they can explicitly impose the required trade-off between accuracy and fairness in the objective function. However, such mechanisms are tightly coupled with the ML algorithm.

Note that the above mentioned papers focused on a centralized setting, in which all information is held by one party, and therefore did not have to deal with privacy considerations.

2.2 Private Collaborative Machine Learning

The Setting

We consider a setting in which several parties wish to collaboratively train ML models on data that is distributed among them, while preventing the disclosure of private information. We refer to this setting as *private collaborative machine learning* (PCML). While many methods were proposed in the literature to address this same (or very similar) setting, they are often associated with different (though overlapping) research domains including *privacy-preserving data mining* [3, 4], *privacy-preserving machine learning* [5], *collaborative machine learning* [6, 7], and *federated machine learning* [8].

The manner in which data is distributed among the collaborating parties is typically categorized as either vertical, horizontal or mixed. In a vertical distribution, each party holds all records with a different subset of attributes, whereas in a horizontal distribution each party holds a subset of the records with all attributes [11]. The mixed scenario refers to an arbitrary partition of the data among the collaborating parties.

When analyzing the privacy preservation of the protocol, it is common to distinguish between two types of adversaries: semi-honest and malicious. A semi-honest adversary is a party that follows the protocol correctly, but tries to infer sensitive information of other parties from intermediate messages that are received during the protocol. A malicious adversary, on the other hand, may deviate from the prescribed protocol in an attempt to learn sensitive information on others. Protocols in the semi-honest model offer weaker security than protocols in the malicious model, as they prevent inadvertent leakage of information between the collaborating parties, and are thus useful for application scenarios in which those are the realistic threats [43]. Protocols in the semi-honest model are viewed as a first step towards achieving higher levels of security. On the other hand, achieving security against malicious parties entails significantly higher costs than achieving security against semi-honest parties. Hence, most of the protocols in the malicious model are protocols that are designed for basic functionalities (such as Oblivious Polynomial Evaluation or Private Set Intersection), while the vast majority of studies on applied privacy-preserving data mining or machine learning (e.g. [44–50]) adopt the semi-honest model.

The two main approaches for achieving privacy in the aforementioned setting are *perturbation* and *secure multi-party computation* as we proceed to describe.

Perturbation

The idea underlying the perturbation approach (e.g. [4]) is to modify the values of attributes randomly, so that privacy will be maintained, while the results of the desired query or analysis on the perturbed data will still be close to the original ones. This technique is at the base of the paradigm called *differential privacy* [9]. An algorithm is considered differentially private if the addition or removal of a single record from the dataset does not significantly affect the output of the algorithm. Typically, differential privacy is obtained by incorporating a Laplacian noise to the results of the data analysis [11].

Application examples of PCML methods that are based on perturbation include classification [5, 51–55], collaborative filtering [56, 57], stochastic gradient descent [58, 59] and deep learning [6, 60–62].

It is important to note that methods that are based on incorporating noise and randomness to the data may perform poorer compared to their non-private counterparts. This problem becomes even more acute in distributed collaborative settings, since in such settings, each party must add noise to the data it holds independently. Such distributed incorporation of noise may result in adding excessive amounts of noise during the training process, and that may yield a considerably inferior model [5, 6].

Secure Multi-Party Computation

In the general setting of *secure multi-party computation* (MPC) [10], L mutually distrustful parties, $P_1, \ldots, P_L$, that hold private inputs, $X_1, \ldots, X_L$, wish to compute some joint function on their inputs, i.e., $f(X_1, \ldots, X_L)$. Ideally, no party should gain during the computation process any information on other parties' inputs, beyond what can be inferred from its own input and the joint function output.

Theoretical results show that such perfect privacy is achievable for any problem of MPC by invoking generic solutions such as Yao's garbled circuit construction [10]. In that approach, one represents the function f as a circuit, either a Boolean one (consisting of `XOR` and `AND` gates) or an arithmetic one (consisting of addition and multiplication gates). The input wires to the circuit bring in the inputs that the parties hold, while the output wires convey the desired output of the computation. Subsequently, the parties run a protocol that emulates the circuit's operation in a secure manner. The circuit-based approach is general and can be used to compute any function on the distributed inputs.

However, as the computational costs are proportional to the size of the circuit, such generic solutions are practical only to rather simple functions. When dealing with more involved functions, such as the ones encountered in ML, such generic solutions become impractical, and more specialized solutions, that are tailored to the function of interest, should be developed. For the sake of higher efficiency, such specialized solutions typically relax the notion of perfect privacy, but do so in a manner that the leaked information is deemed benign.

Application examples of PCML methods that are based on MPC include clustering [63–65], association rule mining [46, 66–68], classification [3, 5, 6, 69–71], and collaborative filtering [49, 72, 73].

2.3 Privacy and Fairness

While many recent studies have investigated algorithms for PCML, the fairness of such algorithms was overlooked. There were, however, several recent studies that incorporated both privacy and fairness considerations to ML algorithms in non-collaborative settings. Most of these studies have investigated the case of a centralized setting, in which all of the dataset is held by a single party

[74–79]. The goal of those studies was to train an ML model over the centralized dataset, making sure that the released model and its future outputs are fair, as well as private, in the sense that one cannot infer from them (meaningful) information about individual data records of the dataset.

A few other studies have investigated a distributed setting which is non-collaborative in the following sense. Their distributed setting includes a "main" party that wishes to train a fair ML model over the data it holds, and a third party to which the sensitive attributes are outsourced. The motivation behind this setting is that while the sensitive attributes should not be exposed to the main party, they should still be used in the training process of the ML model (in a privacy-preserving manner) to ensure that the resulting model is fair. To obtain this goal, these studies used either random projections [80, 81] or MPC techniques [82].

There are few other studies that combine fairness with federated learning. These studies dealt with in-process fairness mechanisms in horizontal federated learning with a single server, in which additional fairness constraints are introduced into ML training optimization procedures [83–88]. These methods are limited in their applicability to only certain ML algorithms, mainly those with non-convex loss functions [83, 84]. Moreover, some of these methods can leak private information due to the need to share excessive information with the server that coordinates the computation [85–87]. An exception in this line of work is the study of Liu et al. [89], that focuses on vertical federated learning. They propose a method to train a fair learning model in the vertical setting in a manner that balances between fairness, accuracy and privacy.

In this paper, we propose a collaborative private pre-process fairness-enhancing mechanism based on MPC, that does not require any third party or server. As a pre-process mechanism, our method is not limited to a specific algorithm, and therefore it can be used with any collaborative ML algorithm.

A concluding remark. The term "fairness" is used in the literature also in another meaning, that inherently differs from *algorithmic* fairness that we consider herein. In algorithmic fairness the goal is to mitigate unintentional biases in the computed machine learning model for the sake of future individuals on which the learnt model will be applied. In contrast, the other meaning of "fairness" is concerned with the interests of the data owners who contribute data to the collaborative learning process and what they expect to get from that collaboration. In cryptography, an MPC protocol is considered fair if it ensures that either all parties receive their designated outputs, or none of them does [11]. In federated learning, it relates to what the data owners, who contribute data of varying quantity and quality and have different computational and communication resources, expect to receive in return from the data federation that they join [90].

3 PROBLEM STATEMENT AND PRELIMINARIES

Let W be a population consisting of n individuals, and let A be a set of *attributes* or *features* that relate to each of those individuals. We considers tabular datasets D that describe the individuals in W in the following manner: each row in D relates to a single individual in W, while each column in D relates to a single attribute in A. The (i, j)-entry in D equals the value of the jth attribute/feature of the ith individual.

We are interested in the case where D is distributed horizontally among L parties, P_ℓ, $\ell \in [L] := \{1, \ldots, L\}$. Namely, party P_ℓ holds a subset of the rows in D, which we denote by D_ℓ, and the subsets are disjoint (in the sense that each individual record appears exactly in one of those partial datasets). They wish to engage in a *fair* collaborative ML classification process over the *entire* dataset

$D = \biguplus_{\ell\in[L]} D_\ell$[1] without disclosing to other parties information on the confidential information (D_ℓ, $\ell \in [L]$) that they hold, beyond what can be naturally deduced from the computed classifier.

In this study we focus on pre-process mechanisms for enhancing fairness in such a distributed setting (as opposed to in-process or post-process, as described in Section 2.1). We consider two adversarial models: one in which the parties are semi-honest, and another in which the parties are malicious (see Section 2.2).

We proceed to introduce the basic notations that will be used throughout the paper. We distinguish between three types of attributes (columns):

• S represents a sensitive attribute (e.g., race or gender). In this work we focus on a binary attribute S that attains one of two possible values, $S \in \{U, V\}$, where U means *unprivileged* and V means *privileged*. (By abuse of notation, we are using S to denote the attribute, as well as its values in different rows of the dataset.)

• X stands for the collection of all non-sensitive attributes. To simplify our discussion, we shall assume that X consists of a single attribute. When X consists of several non-sensitive attributes, we will apply the same pre-process mechanism that we describe below on each such attribute, independently. We shall also assume hereinafter that X is a numerical attribute.

• Y is the binary class that needs to be predicted (e.g. "hire/no hire").

The set of rows, or W, is split in two different manners. The first split is as induced by the sensitive attribute S. Namely, $W = W^U \biguplus W^V$, where W^U is the subset of all individuals in W that are unprivileged (i.e., $S = U$ for them) and $W^V = W \setminus W^U$ is the complementary set of privileged individuals. For each $S \in \{U, V\}$ we let $n^S := |W^S|$.

The other manner in which W is split is according to the distribution of the records of D among the L parties. Namely, $W = \biguplus_{\ell\in[L]} W_\ell$, where W_ℓ is the subset of individuals whose information is held by the party P_ℓ, $\ell \in [L]$. For each $\ell \in [L]$ we let $n_\ell := |W_\ell|$.

Finally, we let n_ℓ^S denote the size of $W_\ell^S := W^S \bigcap W_\ell$, namely, the number of individuals in W^S whose records are held by P_ℓ, $S \in \{U, V\}$, $\ell \in [L]$. Then:

$$n = \sum_{S\in\{U,V\}} n^S = \sum_{\ell\in[L]} n_\ell = \sum_{S\in\{U,V\}} \sum_{\ell\in[L]} n_\ell^S . \tag{1}$$

We shall adopt these superscript and subscript conventions hereinafter. Namely, a superscript S will denote a restriction to the subset W^S, $S \in \{U, V\}$; a subscript ℓ will denote a restriction to the subset W_ℓ, $\ell \in [L]$; a combination of the two will denote a restriction to W_ℓ^S; and no superscript S or subscript ℓ mean that we relate to the entire population W.

In addition, we let $D(X)$ denote the collection of all values appearing in the X-column of D. Similarly, $D^S(X)$, $D_\ell(X)$ and $D_\ell^S(X)$ denote the collection of all values appearing in the X-column of D, restricted to the rows in W^S, W_ℓ or W_ℓ^S, respectively. We assume that $D_\ell^S(X)$, for any S and ℓ, are multisets; namely, they may contain repeated values.

4 THE PROPOSED METHOD

In this section we propose a private pre-process fairness-enhancing mechanism based on MPC for the problem described in Section 3. In Section 4.1 we introduce our pre-process fairness-enhancing mechanism. We do that for a centralized setting, in which all information is held by one party (i.e., $L = 1$); in such a setting privacy is irrelevant. Then, in Section 4.2 we show how to implement that fairness mechanism in the (horizontally) distributed setting, $L > 1$, in a manner that offers privacy to the interacting parties P_ℓ, $\ell \in [L]$. Our protocol invokes a general purpose MPC sub-protocol for computing the kth-ranked element in distributed databases which we describe in Section 4.3.

[1] $\biguplus$ denotes a *disjoint* union

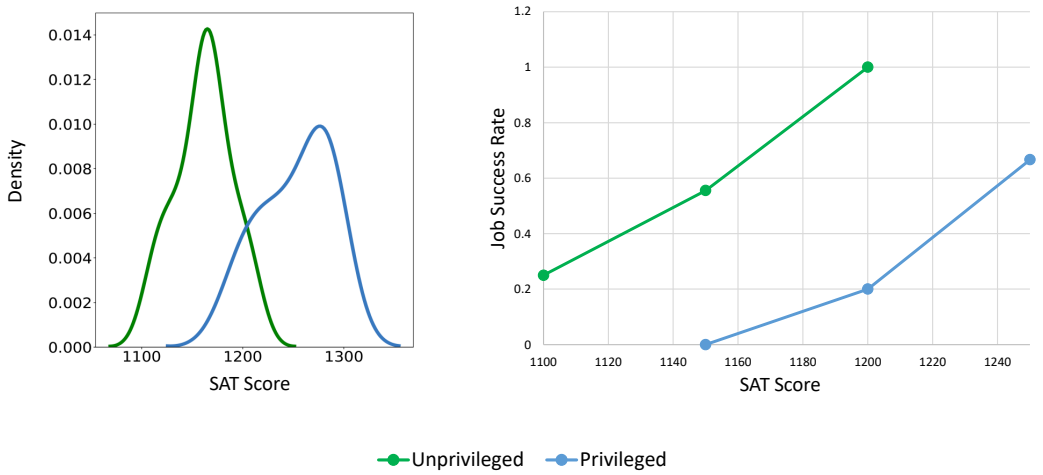

Fig. 1. The distribution of SAT scores within each sub-population (left), and the success rate as a function of the SAT score, within each sub-population (right).

In Section 4.4 we discuss the privacy guarantees of our fairness enhancing mechanism, and we conclude with a communication and computational cost analysize in Section 4.5.

4.1 A Pre-Process Fairness-Enhancing Mechanism

Inspired by [16], we devise a methodology that improves fairness through decreasing distances between the distributions of attributes of the privileged and unprivileged groups, in a pre-process stage. The goal in performing such a repair is to reduce the dependency of the ML model on the group to which an individual belongs, even when that dependency is not a direct one but rather an indirect one through proxy variables. That is, we obfuscate the ability to differentiate between groups using the presumably legitimate non-sensitive variables. In contrast to [16], our method is designed specifically to be suitable for privacy-preserving enhancements, as we do in Section 4.2.

To illustrate the intuition behind the proposed method, consider the example depicted in Figure 1. The figure illustrates a case in which SAT scores of individuals are used to predict their success (or failure) in a given job. The plot on the left shows the distribution of SAT scores, within the two sub-populations in the dataset — the privileged group and the unprivileged group. The plot on the right shows the job success rate as a function of the SAT score, within each of those two sub-populations.

In this example, SAT scores may be used to predict the success (or failure) of candidates in a given job, since the higher the SAT score is, the higher is the probability for success. However, relying solely on the SAT scores, while ignoring the group to which the candidates belong, may create an undesired bias. More specifically, as can be seen from the figure, unprivileged candidates with SAT scores of approximately 1100 perform just as well as privileged candidates with SAT scores of 1200 (e.g., since they may have encountered harder challenges along their way for achieving their scores). Therefore, if SAT scores were used for hiring, for example by just placing a threshold, unprivileged candidates with high potential would be excluded, whereas lower potential candidates from the privileged group would be hired instead. The goal of the pre-process mechanism that we suggest herein is to rectify this bias.

For each group $S \in \{U, V\}$, we first partition the multiset of values in $D^S(X)$ into (nearly) equal-sized bins of sequential values. Namely, if we let B denote the number of bins, then we partition $D^S(X)$ into $D^S(X) = \biguplus_{i=1}^{B} b^S_{(i)}$, where (a) for all $i \in [B-1]$, $y \in b^S_{(i)}$ and $y' \in b^S_{(i+1)}$, we have $y \leq y'$[2], and (b) the bins are of (nearly) equal sizes, in the sense that the sizes of the bins (viewed as multisets) are either $\lfloor n^S/B \rfloor$ or $\lceil n^S/B \rceil$. These two conditions simply state that the resulting bins are the B-quantiles of $D^S(X)$. Below we provide the precise manner in which those bins/quantiles are computed.

After completing the binning process, we proceed to compute the minimal values in each of the B bins, for each group $S \in \{U, V\}$,

$$m^S_{(i)} := \min\{b^S_{(i)}\}, \quad S \in \{U, V\}, \; i \in [B], \tag{2}$$

and also the maximal value in the last bin,

$$m^S_{(B+1)} := \max\{b^S_{(B)}\}, \quad S \in \{U, V\}. \tag{3}$$

Those values enable us to perform a repair of $D^V(X)$, as follows: for every bin $b^V_{(i)}$ in $D^V(X)$, $i \in [B]$, we shift all values in it "towards" the values in the corresponding bin in $D^U(X)$, according to the repair rule described in Eq. (4) below.

$$\bar{x} = (1-\lambda) \cdot x + \lambda \cdot \left(m^U_{(i)} + \left(\frac{x - m^V_{(i)}}{m^V_{(i+1)} - m^V_{(i)}} \right) \cdot \left(m^U_{(i+1)} - m^U_{(i)} \right) \right), \; i \in [B], x \in b^V_{(i)}. \tag{4}$$

Here, x is an original value extracted from the bin $b^V_{(i)}$, while $\bar{x}$ is its repaired value. This repair procedure represents a linear mapping that performs min-max scaling of the values in each bin of the privileged group to the range of values in the corresponding bin in the unprivileged group. The computation brings the distributions of both groups closer.

The repair tuning parameter $\lambda \in [0, 1]$ controls the strength of the repair. If $\lambda = 0$ then we perform no repair, since then $\bar{x} = x$. If $\lambda = 1$ then we get a full repair, since then all values in $b^V_{(i)}$ are replaced with values in the range of the corresponding bin $b^U_{(i)}$, while keeping a similar distribution as the original ones in $b^V_{(i)}$. Note that if $b^V_{(i)}$ elements are all the same, then both the numerator and denominator in the fraction on the right hand side of Eq. (4) are zero. In such a case we interpret that fraction as $1/2$. Such a setting implies that all the (equal) values in $b^V_{(i)}$ will be mapped to the same value in the middle of the range of the corresponding bin $b^U_{(i)}$, namely $\left(m^U_{(i+1)} + m^U_{(i)} \right)/2$.

At the completion of the repair pre-process procedure, an ML model is trained with the repaired dataset. The steps of the above described method are illustrated in Figure 2.

To illustrate the effects of the two main parameters that our mechanism uses, i.e., λ and B, recall the example presented in Figure 1. Figure 3 shows the effect of varying λ values. The higher is the value of λ (with a fixed number of bins, $B = 3$), the closer the distributions are.

Figure 4 shows the effect of varying B values. The figure shows that higher value of B (with a fixed value of $\lambda = 0.9$), yield closer distributions.

In both cases, making the distributions closer means that the two groups will be treated more similarly by the ML model, and, hence, making it harder for the model to distinguish between the two groups. In other words, when using higher values of λ and B, the model will be less likely to make decisions that depend on the group through other presumably legitimate non-sensitive attributes.

[2]We use subscripts in parentheses for any indexing purpose that does not relate to the distribution between the L parties.

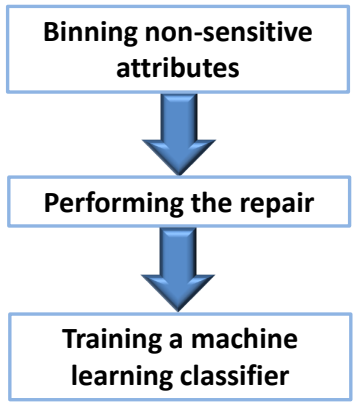

Fig. 2. General steps of the proposed method

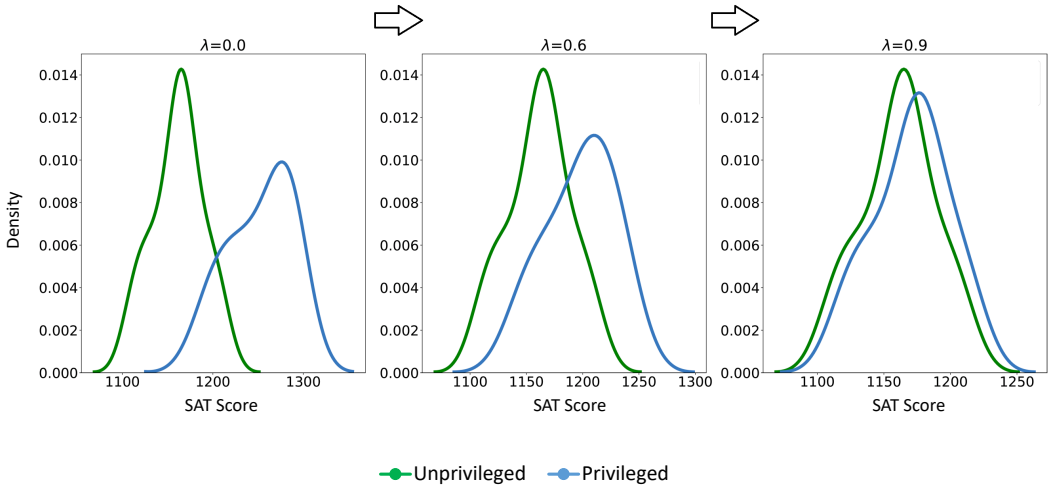

Fig. 3. The effect of the repair tuning parameter λ

To conclude this section, we proceed to provide the formal details of the above-described binning scheme. Let us first order all values in $D^S(X)$, the size of which is n^S, in a non-decreasing manner, as follows:

$$D^S(X) = (x_1, \ldots, x_{n^S}), \quad \text{where } x_1 \le x_2 \le \cdots \le x_{n^S}. \tag{5}$$

Next, we define a set of indices in the ordered multiset $D^S(X)$ in the following manner:

$$K^S := \{k_{(0)} := 0 < k_{(1)} < \cdots < k_{(B-1)} < k_{(B)} := n^S\} \tag{6}$$

Assume next that $n^S = q^S \cdot B + r^S$, where $r^S \in [0, B)$ (i.e., q^S and r^S are the quotient and remainder, respectively, when dividing n^S by B). Then the sequence of indices $k_{(i)}$, $i \in [B]$, is defined by the following equation,

$$k_{(0)} = 0\,; \quad k_{(i)} = k_{(i-1)} + \Delta, \text{ where } \Delta = \begin{cases} q^S + 1 & \text{if} \quad i \le r^S \\ q^S & \text{if} \quad i > r^S \end{cases}, \; i \in [B]\,. \tag{7}$$

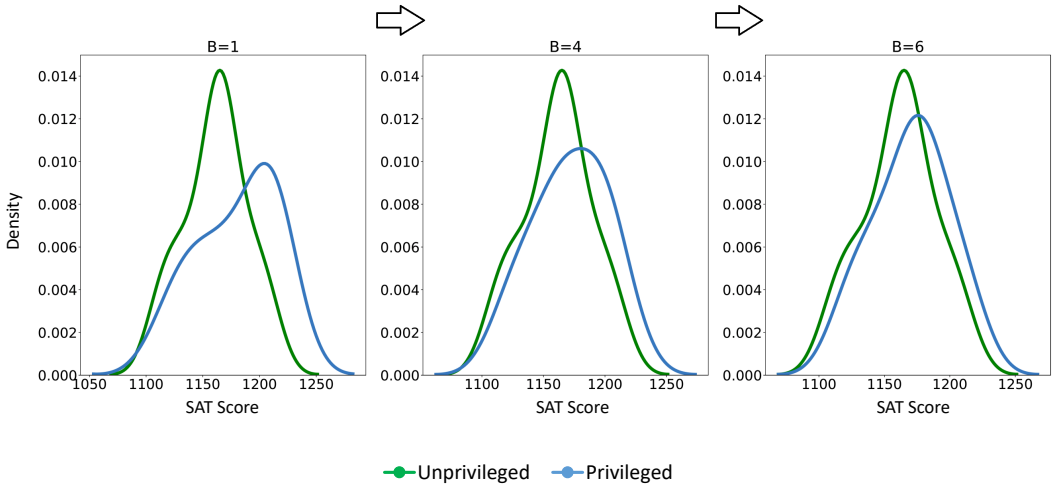

Fig. 4. The effect of the number of bins B

It is easy to verify that $k_{(B)} = n^S$. Finally, the bins in $D^S(X)$, Eq. (5), are defined by K^S as follows:

$$b^S_{(i)} := \{x_j : k_{(i-1)} + 1 \le j \le k_{(i)}\}, \quad i \in [B]. \tag{8}$$

We see, in view of Eq. (8), that the size of the first r^S bins is $q^S + 1$, while all other bins are of size q^S. Moreover, $m^S_{(i)} = \min\{b^S_{(i)}\}$ (Eq. (2)) equals the $(k_{(i-1)} + 1)$-th ranked element in $D^S(X)$, for all $i \in [B]$ and $S \in \{U, V\}$, while $m^S_{(B+1)} := \max\{b^S_{(B)}\}$ (Eq. (3)), equals the maximal element in $D^S(X)$, $S \in \{U, V\}$,

4.2 Secure Multi-Party Algorithm for Enhancing Fairness

In section 4.1 we described our fairness-enhancing mechanism in a centralized setting, i.e. when all of D is held by a single party. We now revisit that algorithm and devise a secure implementation of it, given that the dataset D is horizontally distributed among L parties, as described in Section 3.

To that end, let us go back to Figure 2. The step that poses a challenge when privacy is of concern is the first one: dividing the non-sensitive attribute into equal-sized bins for each group $S \in \{U, V\}$, i.e., $D^S(X) = \biguplus_{i=1}^{B} b^S_{(i)}$, and then compute the boundaries of those bins, $m^S_{(i)} := \min\{b^S_{(i)}\}$ and $m^S_{(B+1)} := \max\{b^S_{(B)}\}$, for $S \in \{U, V\}$ and $i \in [B]$, see Eqs. (2)-(3). Performing those computations poses a challenge in the distributed setting, since they depend on data that is distributed among the L parties and cannot be shared due to privacy concerns.

The second step in Figure 2 poses no problem since it can be carried out by each party locally, independently of others, once the bin boundaries $m^S_{(i)}$, $i \in [B+1]$, $S \in \{U, V\}$, are computed. Even though in that step the parties do not collaborate, they must agree upfront on λ (the repair tuning parameter), see Eq. (4). Note that at this step, every P_ℓ, $\ell \in [L]$, repairs the values of $D^V_\ell(X)$ that it possesses, but it does not share the repaired values with anyone else.

As for the last step in Figure 2, there the parties need to learn a classifier on distributed data, without sharing that data. Since this is, again, a computational problem that involves all of the distributed dataset, privacy issues kick in. However, for such problems of privacy-preserving distributed ML classification there are existing MPC-based solutions, e.g. [3, 69–71, 91].

Protocol 1 provides a pseudo-code that summarizes our mechanism for fairness-driven collaborative machine learning. It begins with the L parties computing in a secure manner the sizes of the two distributed datasets, D^U and D^V (Line 1). They do so without revealing information on the sizes of the partial datasets that each party holds; we explain how this computation is carried out in Section 4.3. Then, they agree on the two parameters that control the repair procedure — B and δ (Line 2). Afterwards, they repair the non-sensitive attributes in each of the two groups (Lines 3-8). The main challenge here is to compute the bin boundaries (Line 5). That computation relies on an MPC sub-protocol for computing the kth-ranked element in a distributed dataset. Such a sub-protocol is described in Section 4.3. Finally, the parties perform the desired distributed machine learning computation on the repaired version of the unified dataset (Line 9).

Protocol 1 A fairness-driven protocol for collaborative machine learning.

1: The L collaborating parties, $P_1, \ldots, P_L$, compute the number of individuals in the underlying population W in each of the two groups — n^U and n^V.
2: The parties agree on the number of bins, B, and the repair tuning parameter λ.
3: **for all** group $S \in \{U, V\}$ **do**
4: **for all** non-sensitive attribute X **do**
5: The parties engage in an MPC computation of the $(B+1)$ bin boundaries of the unified multiset $D^S(X)$ (see Eqs. (2)+(3)).
6: Each party P_ℓ, $\ell \in [L]$, repairs the values in $D^V_\ell(X)$ according to Eq. (4) and the computed bin boundaries.
7: **end for**
8: **end for**
9: The parties engage in a distributed machine learning computation over the repaired datasets.

Therefore, we focus in the next section on the problem of privacy-preserving binning of a distributed dataset. That computation consists of two MPC protocols: a simple one for computing the number of entries in a distributed dataset (as done in Line 1 of Protocol 1) and a more involved one for computing the kth-ranked element in such a dataset (as done $(B+1)$ times in Line 5 of Protocol 1). In our discussion of those two MPC sub-protocols we omit the superscript that denotes the group, since the computation is carried out in the same manner for each the two groups.

We conclude by noting that as cryptographic protocols usually operate over finite fields, it is essential to convert all real data into integers before subjecting them to such computations. To that end we apply the following simple procedure to convert real values into integers (with some precision loss), prior to executing the protocol. If we are interested in preserving a precision of d digits after the decimal point, we multiply each real value by 10^d and round the resulting value to the nearest integer. After executing our mechanism, we divide the values by 10^d.

4.3 A protocol for computing the kth-ranked element in distributed datasets

4.3.1 Overview. In our setting there are L parties, P_ℓ, $\ell \in [L]$, each one holding a *private* dataset D_ℓ that consists of n_ℓ scalars. Let $n = \sum_{\ell \in [L]} n_\ell$ and let $k \in [n]$. The parties wish to compute in a *secure* manner the kth-ranked element in $D = \bigcup_{\ell \in [L]} D_\ell$; namely, the value $x \in D$ such that exactly $k-1$ elements in D are less than or equal to x while $n-k$ elements in D are greater than or equal to x. Protocol 2 that is described below solves that problem. That is the protocol which is called $B+1$ times in Line 5 of Protocol 1 in order to compute the kth-ranked elements that constitute the boundaries of the B bins.

The $B+1$ relevant values of k are described in Section 4.1, see the last paragraph there and Eq. (7). As we can see, those values of k depend on B, as well as on $n = \sum_{\ell \in [L]} n_\ell$. We, however, assume that the sizes of the private datasets, n_ℓ, $\ell \in [L]$ are kept secret. Below we clarify how the parties securely compute n, without leaking any further information on n_ℓ, $\ell \in [L]$.

Protocol 2, which we describe in the following, is based on the protocol template that was offered by Aggarwal et al. in [92]. That protocol template left some MPC components unspecified, and we suggest in Protocol 2 specific implementations of those components. Moreover, the protocol in [92] assumed that the sizes of the private datasets, n_ℓ, $\ell \in [L]$, are public. Our Protocol 2 keeps those values private.

The main cryptographic shield on which Protocol 2 bases its security is secret sharing [93]. Specifically, each party P_ℓ, $\ell \in [L]$, secret shares its private information (which are counts of records in its own private dataset D_ℓ) among all parties using Shamir's secret sharing scheme with the threshold set to

$$t = \lfloor (L+1)/2 \rfloor . \tag{9}$$

All subsequent computations are carried out on those secret shares by invoking suitable MPC sub-protocols. With such a setting of the threshold, the protocol is secure under the assumption of an *honest majority*; namely, that if some of the parties collude in attempt to recover the secret shared information then their number is less than $L/2$.

Protocol 2 is designed to deal with both semi-honest and malicious parties (see Section 2.2 for the definition of those notions). Specifically, Protocol 2 in its entirety solves the problem in a secure manner for the malicious case, while if we remove from it Lines 3, 10, 15, and 18 that are marked by [Mal.], the protocol solves the problem securely in the semi-honest case.

4.3.2 In preparation for running the protocol. Before running Protocol 2, each party P_ℓ secret shares n_ℓ among all parties using t-out-of-L Shamir's threshold secret sharing scheme, with t as in Eq. (9). Then, each of the parties adds up all shares that it got in n_ℓ for all $\ell \in [L]$. Owing to the linearity of secret sharing, the parties obtain that way a t-out-of-L secret sharing of $n = \sum_{\ell \in [L]} n_\ell$. Now, they can publicly use any t of those shares in order to reconstruct n. Hence, n_ℓ, $\ell \in [L]$, are kept secret, but n is computed and made public. That is the computation that the parties perform in Line 1 of Protocol 1.

In addition, the parties agree on the number of bins B. Once n and B are known, the sequence of $B+1$ ranks k that determine the bin boundaries can be computed. (See Line 2 of Protocol 1.)

Finally, the parties jointly determine apriori lower and upper bounds, α and β, on all elements of D. Now all is ready for the execution of Protocol 2 (which is executed in Line 5 of Protocol 1).

4.3.3 Semi-honest parties. We begin by describing the protocol for the semi-honest case; namely, we ignore for now the lines marked by [Mal.].

The protocol computes the kth-ranked element by implementing a privacy-preserving binary search. The lower and upper bounds of the search interval, denoted a and b, are initialized in Line 1. A Boolean flag called 'found' is set to **false** in Line 2; it will be set to **true** once the kth-ranked element is found and then the search will stop.

The main loop takes place in Lines 4-20. First, m is publicly set by all parties to the middle of the current search interval (Line 5). Then, each party counts how many elements in its own private dataset are smaller than m and how many are larger than m; those two private counters are denoted x_ℓ and y_ℓ, respectively (Line 7). Subsequently, each party P_ℓ performs secret sharing of those two private counters among all parties (Line 8); as stated earlier, the secret sharing scheme is Shamir's t-out-of-L scheme with t as in Eq. (9).

Protocol 2 Computing the kth-ranked element in a distributed dataset.

1: Set $a = \alpha$, $b = \beta$.
2: Set found = **false**.
3: [Mal.] Set $X_\ell = Y_\ell = 0$ for all $\ell \in [L]$.
4: **repeat**
5: Set $m \leftarrow \lceil (a+b)/2 \rceil$.
6: **for all** $\ell \in [L]$ **do**
7: P_ℓ sets $x_\ell = |\{u \in D_\ell : u < m\}|$ and $y_\ell = |\{u \in D_\ell : u > m\}|$.
8: P_ℓ secret shares x_ℓ and y_ℓ among all parties.
9: **end for**
10: [Mal.] For each $\ell \in [L]$ the parties securely verify that $x_\ell + y_\ell \le n_\ell$, $x_\ell \ge X_\ell$, and $y_\ell \ge Y_\ell$.
11: **if** $\sum_{\ell \in [L]} x_\ell \le k-1$ and $\sum_{\ell \in [L]} y_\ell \le n-k$ **then**
12: Set found = **true**.
13: **else if** $\sum_{\ell \in [L]} x_\ell \ge k$ **then**
14: Set $b = m - 1$.
15: [Mal.] Set $Y_\ell = n_\ell - x_\ell$.
16: **else if** $\sum_{\ell \in [L]} y_\ell \ge n - k + 1$ **then**
17: Set $a = m + 1$.
18: [Mal.] Set $X_\ell = n_\ell - y_\ell$.
19: **end if**
20: **until** found
21: Output m.

Then (Line 11), the parties engage in an MPC sub-protocol to verify two inequalities. We defer to a later stage the details of that sub-protocol. If both inequalities are verified then the current value of m is the sought-after kth-ranked element. Therefore, we set the value of the Boolean flag found to **true** so that the **repeat** loop will stop and issue the current value of m as the output.

Otherwise, the parties securely verify the inequality in Line 13. If it is verified then the overall number of elements in the unified dataset D that are smaller than m is at least k. In that case we can safely deduce that the value of the kth-ranked element is smaller than m. Hence, we modify the value of the upper bound on the kth-ranked element (Line 14) and stay in the repeat loop for another iteration. Similarly, if the inequality in Line 16 is verified, then the parties update the lower bound in the binary search (Line 17). The search thus proceeds until convergence.

It is easy to see that if all parties are semi-honest the search will converge to the correct value of the kth-ranked element. Furthermore, given that the MPC sub-protocol that is invoked in Lines 11, 13 and 16 is perfectly secure (namely, that it only outputs a single bit that indicates whether the inequality holds or not, and nothing beyond that), the parties only learn the sequence of updated lower and upper bounds, a and b. However, such information is implied by the computed output and the apriori bounds α and β. Hence, the protocol does not leak any information to the interacting parties apart from the computed output.

4.3.4 Malicious parties. We now turn our attention to the enhancements in Protocol 2 for the sake of addressing potential malicious conduct. A malicious party P_ℓ, $\ell \in [L]$, could sabotage the search by submitting inconsistent values of x_ℓ or y_ℓ. Such a sabotage could prevent the convergence of the protocol, or divert it to a wrong output. The added lines in the protocol offer a layer of protection against such malicious conduct. Those enhancements to the protocol introduce two new counters that are secret shared among all parties: X_ℓ, that equals, in every stage of the protocol's run, the

number of elements in D_ℓ that are smaller than a, and Y_ℓ that equals the number of elements in D_ℓ that are greater than b, $\ell \in [L]$.

Those two counters are initialized to zero (Line 3) since initially a and b are lower and upper bounds on all elements in D_ℓ for all $\ell \in [L]$. When the search interval is shrank to its lower half (Lines 13-15) only Y_ℓ has to be updated. Its new value is set to $n_\ell - x_\ell$, since x_ℓ equals the number of elements in D_ℓ that are less than or equal the new upper bound b. Similarly, if the search interval is shrank to its upper half (Lines 16-18), X_ℓ is updated to equal $n_\ell - y_\ell$.

The three verifications that are carried out in Line 10 check that all information that was provided by P_ℓ is consistent. Indeed, $x_\ell + y_\ell$ is supposed to be the number of elements in D_ℓ that are strictly smaller or larger than m, and hence it should be at most n_ℓ, which is the size of D_ℓ that P_ℓ had reported in preparation for running this protocol through the secret shares that it distributed (see Section 4.3.2). Since $a \leq m$ then $\{u \in D_\ell : u < a\} \subseteq \{u \in D_\ell : u < m\}$, and therefore we should have $x_\ell \geq X_\ell$. Same arguments apply for the third inequality that is verified in Line 10, namely, $y_\ell \geq Y_\ell$. It can be shown that the verification of those three inequalities suffices. Namely, if the protocol ends successfully and outputs some value m, then the values provided by P_ℓ, for any $\ell \in [L]$, during the protocol's run are consistent with some dataset D_ℓ of size n_ℓ, so that m is the kth-ranked element in $D = \bigcup_{\ell \in [L]} D_\ell$. (See [92, Theorem 5]).

4.3.5 An MPC sub-protocol for verifying inequalities. Each of the four inequalities in Lines 11, 13 and 16 is equivalent to an inequality of the form $v > 0$, where v is an integer in which the parties hold t-out-of-L secret shares. Furthermore, in all those inequalities v is in the range $[-n, n]$. Let us assume that the field $\mathbb{Z}_p$ that underlies the secret sharing scheme is sufficiently large, and in particular that $p > 2n$.

Lemma 4.1. *Let $n > 0$ and v be two integers so that $v \in [-n, n]$ and let $p > 2n$ be a prime. Then $v > 0$ if and only if the LSB of $(-2v \mod p)$ is 1.*

Proof. Assume that $v > 0$, namely, that $v \in (0, n]$. Hence, $-2v \in [-2n, 0)$. Therefore, as $2n < p$, $(-2v \mod p) = -2v + p$. As that number is odd, its LSB is 1. If, on the other hand, $v \leq 0$, then $v \in [-n, 0]$. Hence, $-2v \in [0, 2n] \subset [0, p-1]$. Therefore, $(-2v \mod p) = -2v$. As that number is even, its LSB is 0. □

Hence, in order to check that $v > 0$, the parties need to check that the LSB of $(-2v \mod p)$ is 1. To that end, Nishide and Ohta [94] suggested an MPC protocol for computing the LSB of a secret shared value in a secure manner.

To illustrate the above course of action, let us consider the inequality in Line 11, $\sum_{\ell \in [L]} x_\ell \leq k - 1$. It is equivalent to $v > 0$ where $v = k - \sum_{\ell \in [L]} x_\ell$. As v is a difference between two integers in the range $[0, n]$ we infer that $v \in [-n, n]$. Since each party P_j holds a share $x_\ell[j]$ in x_ℓ, for all $j, \ell \in [L]$, then P_j can compute $v[j] = k - \sum_{\ell \in [L]} x_\ell[j]$ and that would be its share in v. Therefore, $-2v[j]$ would be P_j's share in $-2v$, $j \in [L]$ (all computations are done in $\mathbb{Z}_p$). Those will be the shares that the parties will input to the MPC sub-protocol for computing the LSB of $-2v$, which will indicate whether $v > 0$.

However, in the malicious case we cannot use the above solution since it relies on the assumption that all secret shared values are smaller than or equal to n. Indeed, malicious parties may distribute in Line 8 secret shares in values x_ℓ and y_ℓ where x_ℓ, y_ℓ or $x_\ell + y_\ell$ are greater than n, and then such malicious conduct will sabotage the above suggested method for inequality verification. Hence, in the malicious case, we suggest using another method for verifying inequalities that was suggested in [94]. If u and v are two elements in $\mathbb{Z}_p$ that are secret-shared among $P_1, \ldots, P_L$ using t-out-of-L Shamir's threshold scheme, then that method enables the parties to verify the inequality $u < v$ (when u and v are viewed as nonnegative integers) without learning any further information on

those two numbers. That method does not rely on the assumption that $u, v \in [0, n]$. However, it involves three invocations of the LSB sub-protocol (instead of just one, as described earlier), hence it is more costly. We omit further details of that computation; interested readers are referred to [94].

4.4 Privacy Analysis

Our mechanism, which is summarized in Protocol 1, begins with each party P_ℓ, $\ell \in [L]$, holding its share of the dataset, D_ℓ, and ends with P_ℓ holding the repaired version of D_ℓ. The information that needs to be protected is the original set of records D_ℓ.

The only places during Protocol 1 in which the parties exchange messages relating to their private datasets is in Lines 1 and 5. In Line 1 the parties perform a simple t-out-of-L secret sharing of their private counters n_ℓ (see Section 4.3.2). As the value of the threshold t is set in Eq. (9) to be greater than or equal to $L/2$, such secret sharing offers perfect security under the assumption of honest majority.

As for Protocol 2 that is invoked in Line 5 of Protocol 1, both variants of that protocol, for the semi-honest and for the malicious case, are perfectly secure, as argued in Sections 4.3.3 and 4.3.4. Namely, the parties are unable to extract from their views during the execution of that protocol anything beyond the computed output and what can be inferred from that output and their own input on the private data of other parties. Hence, the only leakage of information in Protocol 1 is the $B + 1$ bin boundaries for each of the two groups, $S \in \{U, V\}$.

The bin boundaries allow the parties to infer some excessive information on the private data of their peers. To illustrate the type of such inferences, let us consider one party P_ℓ and bin number $i \in [B]$, in which all values are within the interval $I := [m_{(i)}, m_{(i+1)}]$. The size of the bin is also publicly known and it equals $\lfloor n/B \rfloor$ or $\lceil n/B \rceil$. Hence, since P_ℓ knows how many entries in its own dataset are within the interval I, it may infer the number of elements in $D \setminus D_\ell$ in I. However, P_ℓ cannot make any specific inferences about any of the other parties, as it cannot tell how those elements are spread between the other private datasets, nor can it link those records to specific individuals. Hence, such an information leakage is arguably benign.

We observe that the leaked information increases when B increases. When $B = 1$, only the minimal and maximal values in $D(X)$ are exposed to the collaborating parties, whereas if $B = n$, all of the values in $D(X)$ are exposed (but still, their allocation to parties or individuals is kept secret). In Section 5, where we evaluate our method, we show that using a small number of bins (e.g., $B = 3$) enhances fairness considerably, while the marginal contribution of higher values of B for fairness-enhancement are insignificant. Hence, it appears that using a small number of bins, such as $B = 3$, is the preferred choice, as it achieves a considerable enhancement of fairness, while it entails very benign information leakages, as discussed above, and reduced communication and computational costs, as it reduces the number of invocations of Protocol 2.

4.5 Computational and Communication Costs

The parameters that determine the computational and communication costs of Protocol 1 are:

- L – number of parties.
- N_X – the number of non-sensitive attributes.
- M – a uniform bound on the range of possible values in each of the non-sensitive attributes. (It is determined by the minimal and maximal values, α and β, in each of the non-sensitive attributes).
- p – the size of the underlying finite field in which all secret sharing takes place. Note that p must be greater than L as well as greater than $n = |D|$. However, for the sake of performing the secure comparison sub-protocol, we must select p to be greater than $2n$ (see Lemma 4.1).

- B – the selected number of bins.

The main operations in Protocol 1 are:

(1) Two secure summation protocols to compute n^U and n^V. Each such protocol involves adding L secret numbers in $\mathbb{Z}_p$ (Line 1).
(2) $(B+1)N_X$ invocations of Protocol 2 (Line 5).

The secure summation protocol, for adding L secret numbers in $\mathbb{Z}_p$, involves the following costs per party: L evaluations of a polynomial of degree $t-1$ over $\mathbb{Z}_p$ and communicating messages of $O(L \log p)$ bits. Finally, it is necessary to perform a single Lagrange interpolation in order to recover the required sum. As those computations are done just once (in Line 1), their effect on the overall cost is negligible.

We therefore turn our attention to analyzing Protocol 2. The number of iterations in the repeat loop is bounded by $\log M$, as it is a binary search over the range $[\alpha, \beta]$, the size of which is bounded by M. In each iteration there are $2L$ operations of secret sharing (Line 8) followed by 2-4 inequality verifications in the semi-honest case (Lines 11,13,16) or 5-7 inequality verifications in the malicious case (Lines 10,11,13,16).

Hence, the main bottleneck is in the inequality verifications. As explained in Section 4.3.5, such verifications can be done by performing a single LSB computation over shared values in the semi-honest case, or three such LSB computations in the malicious case. A single LSB computation of a shared value in $\mathbb{Z}_p$ involves performing $93 \log p + 1$ multiplications of two secret shared field elements [94]. Finally, a single multiplication of two secret shared values requires performing two secret sharings, each of which has a communication cost of $O(L \log p)$ bits per party. The above analysis implies that the overall communication cost is bounded by $O(BN_X L \log M \log^2 p)$. (The computational cost is analyzed similarly but we omit further details as it would require a detailed discussion of the LSB protocol and that is beyond the scope of the present work. Interested readers may refer to [94] for those details.)

4.6 A concluding remark

The fairness enhancing mechanism of [16] is in fact a special case of our mechanism, where the number of bins is set to $B = \min\{n^U, n^V\}$. Namely, it equals the size of the smaller of the two groups — the unprivileged group, W^U, and the privileged one, W^V. Using binning this way totally exposes all records in the smaller group, and reveals too much information on the larger group. Hence, such a course of action cannot be taken when privacy is of concern. Moreover, the computational and communication costs of the mechanism depend linearly on B, as the number of invocations of the costly MPC sub-protocol (Section 4.3) depends linearly on B (see Section 4.5). Finally, as we show in our experimental evaluation (see Section 5), using $B = 3$ bins is sufficient for mitigating unfairness; using larger numbers of bins barely contributes to the decrease in unfairness.

5 EVALUATION

5.1 Experimental Setting

In this section we present the experimental setting in terms of the datasets, the measures we used, and the parameters' space, and we also provide several implementation details.

Datasets

We evaluated the proposed method using several real-world datasets: the publicly available ProPublica Recidivism dataset, the ProPublica Violent Recidivism dataset, and the Bank Marketing dataset.

Table 1. Confusion matrix

		Predicted "N"	Predicted "P"
Actual	**N**	TN	FP
	P	FN	TP

ProPublica Recidivism Dataset. This dataset includes data from the COMPAS risk assessment system (see [13, 95]). The dataset includes 10 attributes, such as the number of previous felonies, charge degree, age, race, gender etc., and it has 6167 individual records. The target attribute indicates whether an individual has recidivated (namely, was arrested again) after two years or not. The sensitive attribute in this dataset is race. All Caucasians constitute the privileged group, while all individuals of other races constitute the unprivileged group. We use the pre-processed files provided by [96].

ProPublica Violent Recidivism Dataset. This dataset (see [13]) is similar to the ProPublica Recidivism dataset mentioned above. It has the same set of 10 attributes, and the sensitive attribute is race. However, this dataset contains 4010 individual records, and the target attribute indicates the recidivism of a *violent* crime within two years. Here as well, we use the pre-processed files provided by [96].

The Bank Marketing Dataset. This dataset contains information on subscriptions to term deposits in a Portuguese banking institution (see [97]). It includes 41,188 individual records with 20 attributes. In our evaluation we used the 10 attributes out of the 20 which are numeric. The target attribute indicates whether an individual has subscribed to a term deposit. The sensitive attribute in this case is the age. Individuals of ages below 25 and above 60 are considered as the unprivileged group (as in [35]). This dataset is available at the UCI repository [98].

Measures

As mentioned, there is an inherent trade-off between prediction performance and fairness. We wish to evaluate how this trade-off is reflected in the application of our method. To that end, consider the confusion matrix shown in Table 1:

Accuracy. Similarly to many studies on algorithmic fairness, we use accuracy as a measure of prediction performance. Accuracy is measured by the proportion of correct classifications, i.e. $(TN + TP)/(N + P)$.

Fairness. For measuring fairness, we consider a measure based on *equalized odds*. As mentioned in Section 2.1, one advantage of this measure (in contrast, for example, to demographic parity), is that a perfectly accurate classifier will be considered fair. More specifically, we define the unfairness measure $\bar{\varphi} := |DFNR| + |DFPR|$, where $|DFNR|$ (resp. $|DFPR|$) is the absolute difference between the FNR (resp. FPR) of the two groups, and FNR (resp. FPR) is the False Negative Rate (resp. False Positive Rate) as defined below:

$$FNR = \frac{FN}{P} = \frac{FN}{FN + TP}, \quad FPR = \frac{FP}{N} = \frac{FP}{FP + TN}.$$

While the equalized odds measure dictates two separate measures, we use the sum of their absolute values in order to obtain a single combined measure. Such a combined measure will allow us an easier examination of the fairness-accuracy trade-off at later stages, where higher values of $\bar{\varphi}$ indicate lower levels of fairness (or higher levels of unfairness).

Note that in the two ProPublica datasets, the value of "recidivated" is considered as the "positive" value, while in the case of the Bank Marketing dataset, a "positive" value indicates that the corresponding individual has not subscribed to a term deposit service.

Distance. To better understand the repairing mechanism of our method, we also measure the distances between the distributions of attributes within the two groups. We do so by computing the *earth mover's distance* (EMD) [99], divided by M (the size of the range of possible values of the attribute, see Section 4.5).

Parameters' Space

For our experiments, we examined varying values of parameters, as follows:

- *Number of bins:* $B \in \{1, 2, 3, 4, 6, 8, 10\}$.
- *Repair tuning parameter:* $\lambda \in \{0.1, 0.2, \ldots, 0.9, 1\}$.
- *Number of parties:* in the majority of our experiments we applied a procedure based on three parties ($L = 3$); only for the sake of measuring runtimes, we used higher numbers of parties ($L \in \{3, 4, 5, 6, 7\}$). Records were distributed among the L parties randomly[3].

The ML classifier in our experiments was Logistic Regression (for experiments with additional classifiers we refer the reader to Appendix B). We used a constant ratio to split the dataset into train set (66.7%) and test set (33.3%); splits were repeated 10 different times in a random manner, and the reported results are the average over these 10 repetitions.

Implementation Details

Our method was implemented in Python assisted with the Virtual Ideal Functionality Framework (VIFF) library for secure multi-party computations (see [100]). As noted above, our method was applied separately on each of the non-sensitive attributes of the considered dataset, prior to the training of an ML model. For our purpose herein – to examine the effects of our pre-process method, we used a simple *non-distributed non-private* implementation of the ML algorithm.

All experiments were executed on a server running Windows Server 2008 R2, having two 6-cores CPU processors (12 virtual CPUs) with a clock speed of 1.9GHz, and 128GB of RAM.

5.2 Results

We first evaluated the effect of our proposed method on unfairness and accuracy. In order to do so, we executed the method using varying values of λ and B, as mentioned in the previous section, and measured the resulting unfairness and accuracy values. To better understand the repairing mechanism of our method, we also measured the resulting distance between the distributions of attributes within the two groups. Recall that accuracy was measured by the proportion of correct classifications, unfairness by $\bar{\varphi}$, and distance by EMD.

Figure 5 shows the effect of the repair tuning parameter λ on the three considered measures for each of the three considered datasets. Each row of charts represents a different dataset: ProPublica Recidivism (top), Propublica Violent Recidivism (center), and Bank Marketing (bottom), while each column of charts represents a different measure: distance (left), unfairness (center) and accuracy (right). In each chart, the X-axis represents λ, while the Y-axis represents the value of the corresponding measure. Colors and line thickness represent the value of B, where thicker lines represent higher values of B. Note that the distances are calculated for each attribute separately

[3]Recall that our privacy enhancing mechanism ensures that the results of executing the fairness-enhancing mechanism are identical to the case of a single non-distributed dataset, and therefore, the manner in which the dataset is distributed does not matter.

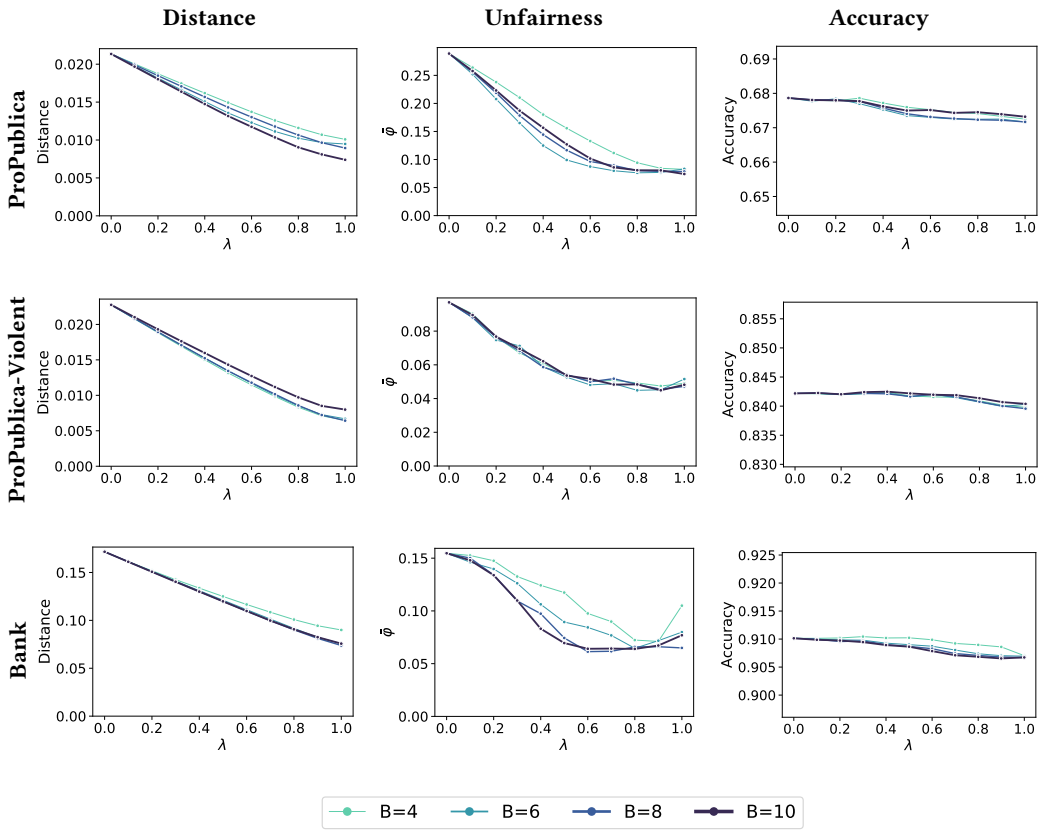

Fig. 5. The effect of the repair tuning parameter λ on distance, accuracy and unfairness. Increasing the value of λ yields a considerable decrease of distance and, consequently, also a decrease of unfairness, with only a minor compromise in accuracy.

and are then averaged over the set of attributes in each dataset. For all three measures, the reported results represent an average over 10 train-test splits.

As can be seen from the figure, by using the proposed method with higher values of λ, it is possible to improve fairness considerably with only a minor compromise in accuracy. The considerable reduction in unfairness is a result of reducing the distances between the distributions of attributes within the two groups, as shown in the left column of charts in the figure. For example, in the case of the ProPublica Recidivism dataset (top row charts), unfairness (top-middle chart) is reduced from 0.29 ($\lambda = 0.0$) to 0.08 ($\lambda = 1.0$) — a reduction by 72%; this is achieved with almost no compromise in accuracy (top-right chart) that is reduced by only 1%. Similar behaviour can be seen with the other datasets.

Figure 6 presents a similar analysis to the one presented in Figure 5 to assess the effect of the number of bins B on the three considered measures. Here, the X-axis of each chart represents the value of B, while the Y-axis represents the value of the corresponding measure. Colors and line thickness represent the value of λ, where thicker lines represent higher values of λ.

As can be seen from the figure, when increasing B, unfairness is reduced with only a minor compromise in accuracy. However, increasing B beyond $B = 3$ has almost no effect on all three

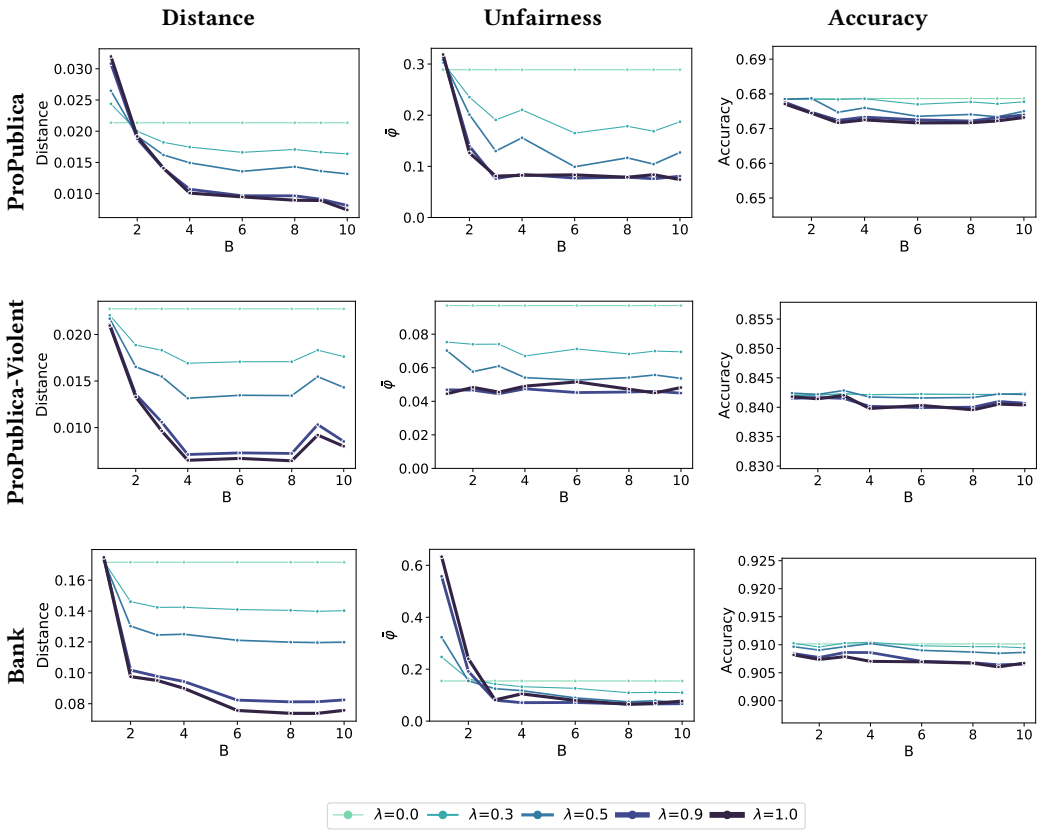

Fig. 6. The effect of the number of bins B on distance, accuracy and unfairness. Increasing the value of B leads to improvement in fairness with only a minor compromise in accuracy. However, increasing B has a diminishing marginal effect, namely, a large number of bins barely contributes to the decrease in unfairness.

measures. In particular, it barely contributes to the decrease in unfairness. For example, in the case of the ProPublica Recidivism dataset (top row of charts), using 10 bins with $\lambda \in \{0.9, 1\}$ obtains about the same results as with only 3 bins.

The latter analysis indicates that it is preferable to use a small number of bins, around $B = 3$. Further increasing the number of bins does not contribute towards enhancing fairness, but it does entail higher computational and communication costs, as well as increased leakage of information, as discussed in Sections 4.4 and 4.5.

We then turned to evaluate the efficiency of our method, as reflected by its runtimes and communication costs. Figure 7 shows the effect of B on the runtime of our method, when considering semi-honest parties and setting their number L to 3. Specifically, we report the runtime for the secure distributed computation of bin boundaries, for both of the groups. Again, each row of charts represents a dataset. In each chart, the X-axis represents B, while the Y-axis represents runtime in minutes. Line colors represent the attributes that were repaired in each dataset. In the legend of each chart we indicate next to each such attribute its range of values. The figure shows that, in accord with the analysis in Section 4.5, the runtime depends linearly on B.

In the next experiment (Figure 8) we tested the effect of L, the number of parties, on the runtime of our method, in both cases of semi-honest (left column of charts) and malicious parties (right

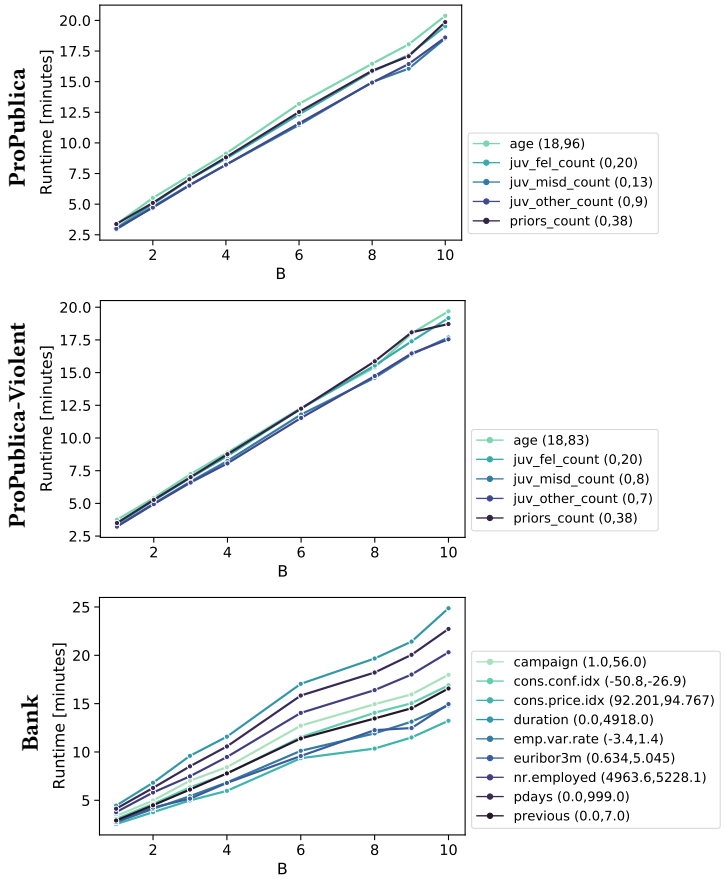

Fig. 7. The effect of the number of bins B on runtime.

column of charts). In each chart, the X-axis represents L, while the Y-axis represents runtime in minutes. Line colors represent the number of bins. For clarity, we present the runtime for repairing one non-sensitive attribute in each of the datasets. For the ProPublica Recidivism and ProPublica Violent Recidivism datasets, we selected the attribute "prior count", while for the Bank Marketing dataset, we selected the attribute "duration". The figure shows that, as indicated by the analysis in Section 4.5, the runtime depends linearly on L.

Finally, we measured the communication costs and report them in Figure 9, which is structured similarly to Figure 8. Here too, the measured communication costs show linear dependence on L, as analyzed in Section 4.5.

In summary, the runtimes and commuication costs of the proposed method, as demonstrated in Figures 7–9, are practical, especially considering that the method is applied once as a pre-process procedure.

To conclude, in the above experiments we showed that our method is able to improve fairness considerably with only a minor compromise in accuracy, despite their inherent trade-off. We further showed that privacy is highly maintained and that the information leakage is minimal, considering

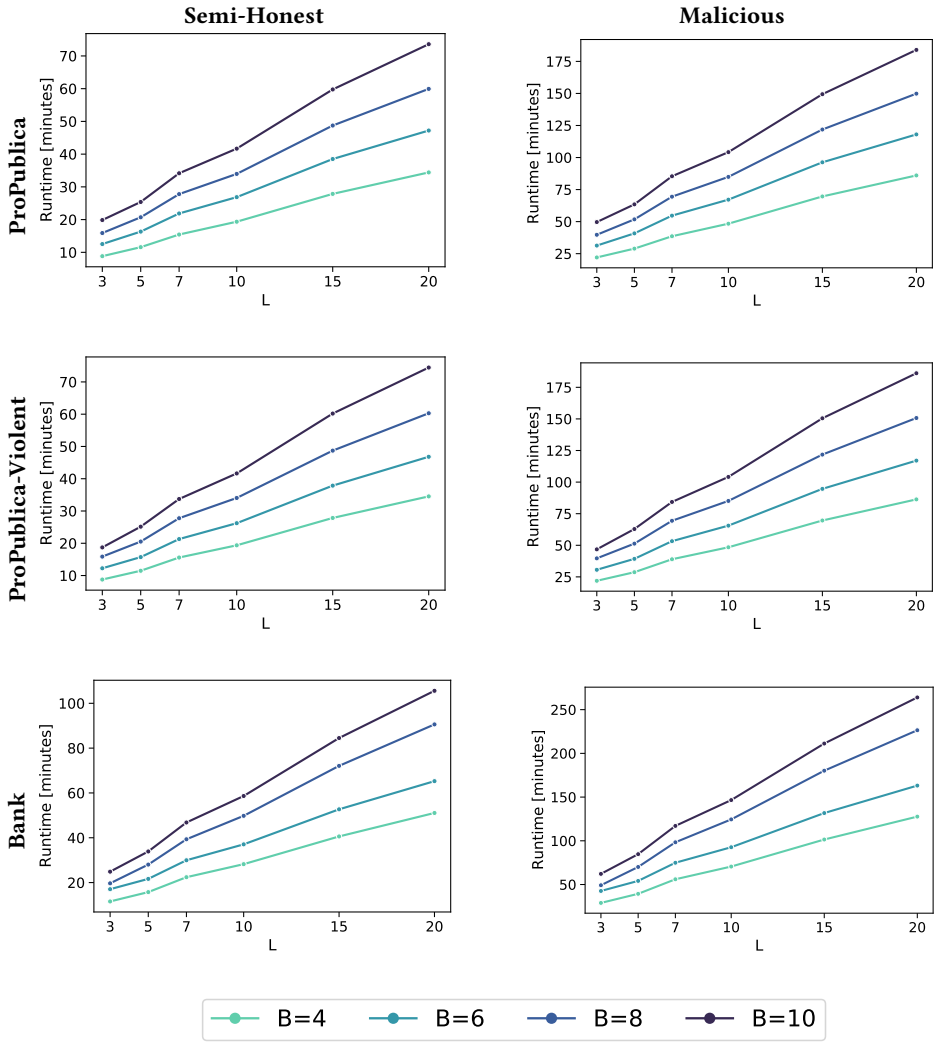

Fig. 8. The effect of the number of parties L on runtime in the case of semi-honest parties (left) and malicious parties (right).

the diminishing effect of the number of bins. Finally, we showed that the runtime of the proposed method is feasible for a one-time pre-process procedure.

6 CONCLUSIONS

In this paper, we proposed a privacy-preserving pre-process mechanism for enhancing fairness of collaborative ML algorithms. In particular, our method improves fairness by decreasing distances between the distributions of attributes of the privileged and unprivileged groups. We use a binning approach that enables the implementation of privacy-preserving enhancements, by means of MPC. Being a pre-process mechanism, our method is not limited to a specific algorithm, and therefore it can be used with any collaborative ML algorithm.

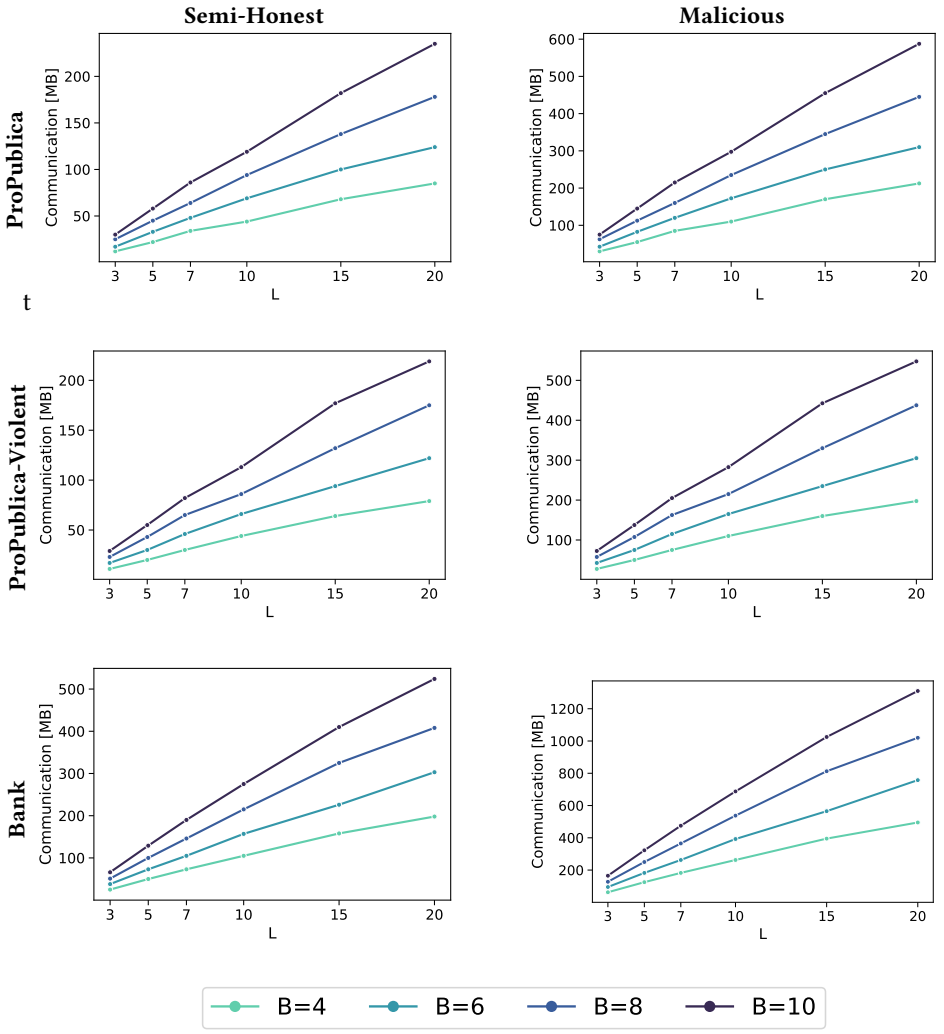

Fig. 9. The effect of the number of parties L on communication costs in the case of semi-honest parties (left) and malicious parties (right).

An extensive evaluation that was conducted using three real-world datasets, revealed that the proposed method is able to improve fairness considerably, with only a minor compromise in accuracy. We also showed that using a small number of bins (e.g., $B = 3$), it is possible to achieve that considerable improvement of fairness, with very minor and benign leakage of information. Finally, we demonstrated that the runtime of the proposed method is practical, especially considering that it is executed once as a pre-process procedure.

Possible future research directions are as follows:

Data distribution. We assumed an horizontal data distribution setting, where each party holds full records of a subset of the population. Forthcoming work may investigate a vertical distribution scenario, where each party holds a subset of the information about all individuals, or a mixed scenario that combines both horizontal and vertical distributions.

Sensitive attribute. We assumed a single sensitive attribute that attains two possible values. While the vast majority of studies in the algorithmic fairness literature makes the same assumptions, future research should consider the case of multiple sensitive attributes that may attain more than just two values.

Fairness mechanism. We proposed a private version of a pre-process fairness enhancing mechanism, inspired by [16]. While it is clear qualitatively how our fairness mechanism's parameters, B and λ, affect fairness, we do not offer theoretical bounds on their effect. Establishing such theoretical bounds could be a promising direction for future research. Moreover, future efforts should be invested in developing private versions of additional fairness enhancing mechanisms, including in-process and post-process mechanisms such as [24, 30, 40].

FUNDING

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.
[2] U.S. Department of Health and Human Services, Summary of the hipaa security rule (1996).
URL https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html
[3] Y. Lindell, B. Pinkas, Privacy preserving data mining, in: Annual International Cryptology Conference, Springer, 2000, pp. 36–54.
[4] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000, pp. 439–450.
[5] B. Jayaraman, L. Wang, D. Evans, Q. Gu, Distributed learning without distress: Privacy-preserving empirical risk minimization, in: Advances in Neural Information Processing Systems, 2018, pp. 6343–6354.
[6] M. Chase, R. Gilad-Bachrach, K. Laine, K. E. Lauter, P. Rindal, Private collaborative neural network learning., IACR Cryptol. ePrint Arch. 2017 (2017) 762.
[7] Y. Hu, D. Niu, J. Yang, S. Zhou, Fdml: A collaborative machine learning framework for distributed features, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2232–2240.
[8] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Transactions on Intelligent Systems and Technology (TIST) 10 (2019) 1–19.
[9] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating noise to sensitivity in private data analysis, in: Theory of cryptography conference, Springer, 2006, pp. 265–284.
[10] A. C. Yao, Protocols for secure computations, in: 23rd annual symposium on foundations of computer science (sfcs 1982), IEEE, 1982, pp. 160–164.
[11] J. Domingo-Ferrer, A. Blanco-Justicia, Privacy-preserving technologies, in: The Ethics of Cybersecurity, Springer, Cham, 2020, pp. 279–297.
[12] W. Du, Y. S. Han, S. Chen, Privacy-preserving multivariate statistical analysis: Linear regression and classification, in: Proceedings of the 2004 SIAM international conference on data mining, SIAM, 2004, pp. 222–233.
[13] J. Angwin, Machine bias — propublica (May 2016).
URL https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing
[14] D. Pessach, E. Shmueli, Algorithmic fairness, arXiv preprint arXiv:2001.09784.
[15] D. Pessach, E. Shmueli, A review on fairness in machine learning, ACM Computing Surveys 55 (3). doi:10.1145/3494672.
URL https://doi.org/10.1145/3494672
[16] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, S. Venkatasubramanian, Certifying and removing disparate impact, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 259–268.
[17] M. J. Zimmer, Emerging uniform structure of disparate treatment discrimination litigation, Georgia Law Review 30 (1995) 563.
[18] S. Barocas, A. D. Selbst, Big data's disparate impact, CALIFORNIA LAW REVIEW 104 (2016) 671.
[19] G. Rutherglen, Disparate impact under title vii: an objective theory of discrimination, Va. L. Rev. 73 (1987) 1297.

[20] J. Kleinberg, S. Mullainathan, M. Raghavan, Inherent trade-offs in the fair determination of risk scores, in: 8th Innovations in Theoretical Computer Science Conference (ITCS 2017), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
[21] S. Verma, J. Rubin, Fairness definitions explained, in: 2018 IEEE/ACM International Workshop on Software Fairness (FairWare), IEEE, 2018, pp. 1–7.
[22] T. Calders, S. Verwer, Three naive bayes approaches for discrimination-free classification, Data Mining and Knowledge Discovery 21 (2010) 277–292.
[23] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, R. Zemel, Fairness through awareness, in: Proceedings of the 3rd innovations in theoretical computer science conference, ACM, 2012, pp. 214–226.
[24] M. Hardt, E. Price, N. Srebro, Equality of opportunity in supervised learning, in: Advances in neural information processing systems, 2016, pp. 3315–3323.
[25] F. Kamiran, T. Calders, Data preprocessing techniques for classification without discrimination, Knowledge and Information Systems 33 (2012) 1–33.
[26] B. T. Luong, S. Ruggieri, F. Turini, k-nn as an implementation of situation testing for discrimination discovery and prevention, in: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2011, pp. 502–510.
[27] C. Louizos, K. Swersky, Y. Li, M. Welling, R. Zemel, The variational fair autoencoder, arXiv preprint arXiv:1511.00830.
[28] F. Calmon, D. Wei, B. Vinzamuri, K. N. Ramamurthy, K. R. Varshney, Optimized pre-processing for discrimination prevention, in: Advances in Neural Information Processing Systems, 2017, pp. 3992–4001.
[29] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, C. Dwork, Learning fair representations, in: International Conference on Machine Learning, 2013, pp. 325–333.
[30] T. Kamishima, S. Akaho, H. Asoh, J. Sakuma, Fairness-aware classifier with prejudice remover regularizer, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2012, pp. 35–50.
[31] B. Woodworth, S. Gunasekar, M. I. Ohannessian, N. Srebro, Learning non-discriminatory predictors, in: Conference on Learning Theory, 2017, pp. 1920–1953.
[32] Y. Bechavod, K. Ligett, Learning fair classifiers: A regularization-inspired approach, arXiv preprint arXiv:1707.00044 (2017) 1733–1782.
[33] Y. Bechavod, K. Ligett, Penalizing unfairness in binary classification, arXiv preprint arXiv:1707.00044.
[34] M. B. Zafar, I. Valera, M. Gomez Rodriguez, K. P. Gummadi, Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment, in: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017, pp. 1171–1180.
[35] M. B. Zafar, I. Valera, M. Gomez Rodriguez, K. P. Gummadi, Fairness constraints: Mechanisms for fair classification, in: Artificial Intelligence and Statistics, 2017, pp. 962–970.
[36] F. Kamiran, T. Calders, M. Pechenizkiy, Discrimination aware decision tree learning, in: 2010 IEEE International Conference on Data Mining, IEEE, 2010, pp. 869–874.
[37] N. Quadrianto, V. Sharmanska, Recycling privileged learning and distribution matching for fairness, in: Advances in Neural Information Processing Systems, 2017, pp. 677–688.
[38] G. Goh, A. Cotter, M. Gupta, M. P. Friedlander, Satisfying real-world goals with dataset constraints, in: Advances in Neural Information Processing Systems, 2016, pp. 2415–2423.
[39] A. Agarwal, A. Beygelzimer, M. Dudik, J. Langford, H. Wallach, A reductions approach to fair classification, in: International Conference on Machine Learning, 2018, pp. 60–69.
[40] S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, A. Huq, Algorithmic decision making and the cost of fairness, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 797–806.
[41] C. Dwork, N. Immorlica, A. T. Kalai, M. Leiserson, Decoupled classifiers for group-fair and efficient machine learning, in: Conference on Fairness, Accountability and Transparency, 2018, pp. 119–133.
[42] A. K. Menon, R. C. Williamson, The cost of fairness in binary classification, in: Conference on Fairness, Accountability and Transparency, 2018, pp. 107–118.
[43] Y. Lindell, B. Pinkas, Secure multiparty computation for privacy-preserving data mining, IACR Cryptol. ePrint Arch. (2008) 197.
[44] G. Asharov, F. Bonchi, D. García-Soriano, T. Tassa, Secure centrality computation over multiple networks, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 957–966.
[45] T. Grinshpoun, T. Tassa, P-syncbb: A privacy preserving branch and bound dcop algorithm, Journal of Artificial Intelligence Research 57 (2016) 621–660.
[46] M. Kantarcioglu, C. Clifton, Privacy-preserving distributed mining of association rules on horizontally partitioned data, IEEE transactions on knowledge and data engineering 16 (2004) 1026–1037.

[47] P. Mohassel, Y. Zhang, Secureml: A system for scalable privacy-preserving machine learning, in: IEEE Symposium on Security and Privacy, 2017, pp. 19–38.
[48] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, S. Moriai, Privacy-preserving deep learning via additively homomorphic encryption, IEEE Trans. Inf. Forensics Secur. 13 (2018) 1333–1345.
[49] E. Shmueli, T. Tassa, Mediated secure multi-party protocols for collaborative filtering, ACM Transactions on Intelligent Systems and Technology (TIST) 11 (2020) 1–25.
[50] S. Zhong, Z. Yang, R. N. Wright, Privacy-enhancing k-anonymization of customer data, in: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2005, pp. 139–147.
[51] K. Chen, L. Liu, Privacy preserving data classification with rotation perturbation, in: Fifth IEEE International Conference on Data Mining (ICDM'05), IEEE, 2005, pp. 4–pp.
[52] O. L. Mangasarian, E. W. Wild, G. M. Fung, Privacy-preserving classification of vertically partitioned data via random kernels, ACM Transactions on Knowledge Discovery from Data (TKDD) 2 (2008) 1–16.
[53] J. Qiang, B. Yang, Q. Li, L. Jing, Privacy-preserving svm of horizontally partitioned data for linear classification, in: 2011 4th International Congress on Image and Signal Processing, Vol. 5, IEEE, 2011, pp. 2771–2775.
[54] K. Chaudhuri, C. Monteleoni, Privacy-preserving logistic regression, in: Advances in neural information processing systems, 2009, pp. 289–296.
[55] M. Pathak, S. Rane, B. Raj, Multiparty differential privacy via aggregation of locally trained classifiers, in: Advances in Neural Information Processing Systems, 2010, pp. 1876–1884.
[56] A. Basu, J. Vaidya, H. Kikuchi, Perturbation based privacy preserving slope one predictors for collaborative filtering, in: IFIP International Conference on Trust Management, Springer, 2012, pp. 17–35.
[57] A. Jeckmans, Q. Tang, P. Hartel, Privacy-preserving collaborative filtering based on horizontally partitioned dataset, in: 2012 International Conference on Collaboration Technologies and Systems (CTS), IEEE, 2012, pp. 439–446.
[58] S. Song, K. Chaudhuri, A. D. Sarwate, Stochastic gradient descent with differentially private updates, in: 2013 IEEE Global Conference on Signal and Information Processing, IEEE, 2013, pp. 245–248.
[59] A. Rajkumar, S. Agarwal, A differentially private stochastic gradient descent algorithm for multiparty classification, in: Artificial Intelligence and Statistics, 2012, pp. 933–941.
[60] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang, Deep learning with differential privacy, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 308–318.
[61] H. B. McMahan, D. Ramage, K. Talwar, L. Zhang, Learning differentially private recurrent language models, in: International Conference on Learning Representations, 2018.
[62] R. C. Geyer, T. Klein, M. Nabi, Differentially private federated learning: A client level perspective, arXiv preprint arXiv:1712.07557.
[63] Z. Erkin, T. Veugen, T. Toft, R. L. Lagendijk, Privacy-preserving distributed clustering, EURASIP Journal on Information Security 2013 (2013) 4.
[64] S. Jha, L. Kruger, P. McDaniel, Privacy preserving clustering, in: European symposium on research in computer security, Springer, 2005, pp. 397–417.
[65] X. Lin, C. Clifton, M. Zhu, Privacy-preserving clustering with distributed em mixture modeling, Knowledge and information systems 8 (2005) 68–81.
[66] T. Tassa, Secure mining of association rules in horizontally distributed databases, IEEE Transactions on Knowledge and Data Engineering 26 (2013) 970–983.
[67] J. Vaidya, C. Clifton, Privacy preserving association rule mining in vertically partitioned data, in: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002, pp. 639–644.
[68] J. Zhan, S. Matwin, L. Chang, Privacy-preserving collaborative association rule mining, in: IFIP Annual Conference on Data and Applications Security and Privacy, Springer, 2005, pp. 153–165.
[69] A. B. Slavkovic, Y. Nardi, M. M. Tibbits, "secure" logistic regression of horizontally and vertically partitioned distributed databases, in: Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), IEEE, 2007, pp. 723–728.
[70] S. Samet, Privacy-preserving logistic regression, Journal of Advances in Information Technology Vol 6.
[71] S. E. Fienberg, W. J. Fulp, A. B. Slavkovic, T. A. Wrobel, "secure" log-linear and logistic regression analysis of distributed databases, in: International Conference on Privacy in Statistical Databases, Springer, 2006, pp. 277–290.
[72] H. Polat, W. Du, Privacy-preserving collaborative filtering on vertically partitioned data, in: European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2005, pp. 651–658.
[73] I. Yakut, H. Polat, Arbitrarily distributed data-based recommendations with privacy, Data & Knowledge Engineering 72 (2012) 239–256.
[74] M. Jagielski, M. Kearns, J. Mao, A. Oprea, A. Roth, S. Sharifi-Malvajerdi, J. Ullman, Differentially private fair learning, in: International Conference on Machine Learning, 2019, pp. 3000–3008.

[75] D. Xu, S. Yuan, X. Wu, Achieving differential privacy and fairness in logistic regression, in: Companion Proceedings of The 2019 World Wide Web Conference, 2019, pp. 594–599.
[76] H. Mozannar, M. I. Ohannessian, N. Srebro, Fair learning with private demographic data, arXiv preprint arXiv:2002.11651.
[77] E. Bagdasaryan, O. Poursaeed, V. Shmatikov, Differential privacy has disparate impact on model accuracy, in: Advances in Neural Information Processing Systems, 2019, pp. 15479–15488.
[78] C. Huang, X. Chen, P. Kairouz, L. Sankar, R. Rajagopal, Generative adversarial models for learning private and fair representations.
[79] R. Cummings, V. Gupta, D. Kimpara, J. Morgenstern, On the compatibility of privacy and fairness, in: Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization, 2019, pp. 309–315.
[80] H. Hu, Y. Liu, Z. Wang, C. Lan, A distributed fair machine learning framework with private demographic data protection, in: 2019 IEEE International Conference on Data Mining (ICDM), IEEE, 2019, pp. 1102–1107.
[81] J. Fantin, A distributed fair random forest (doctoral dissertation), University of Wyoming (2020).
[82] N. Kilbertus, A. Gascón, M. J. Kusner, M. Veale, K. P. Gummadi, A. Weller, Blind justice: Fairness with encrypted sensitive attributes, in: Proceedings of the 35th International Conference on Machine Learning (ICML 2018), Vol. 80, 2018, pp. 2635–2644.
[83] S. Hu, Z. S. Wu, V. Smith, Provably fair federated learning via bounded group loss, arXiv preprint arXiv:2203.10190.
[84] A. Papadaki, N. Martinez, M. Bertran, G. Sapiro, M. Rodrigues, Minimax demographic group fairness in federated learning, arXiv preprint arXiv:2201.08304.
[85] W. Du, D. Xu, X. Wu, H. Tong, Fairness-aware agnostic federated learning, in: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM), SIAM, 2021, pp. 181–189.
[86] Y. Zeng, H. Chen, K. Lee, Improving fairness via federated learning, arXiv preprint arXiv:2110.15545.
[87] L. Chu, L. Wang, Y. Dong, J. Pei, Z. Zhou, Y. Zhang, Fedfair: Training fair models in cross-silo federated learning, arXiv preprint arXiv:2109.05662.
[88] M. Mohri, G. Sivek, A. T. Suresh, Agnostic federated learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 4615–4625.
[89] C. Liu, Z. Zhou, Y. Shi, J. Pei, L. Chu, Y. Zhang, Achieving model fairness in vertical federated learning, arXiv preprint arXiv:2109.08344.
[90] Y. Shi, H. Yu, C. Leung, A survey of fairness-aware federated learning, CoRR abs/2111.01872.
URL https://arxiv.org/abs/2111.01872
[91] H. Kikuchi, H. Yasunaga, H. Matsui, C.-I. Fan, Efficient privacy-preserving logistic regression with iteratively re-weighted least squares, in: 2016 11th Asia Joint Conference on Information Security (AsiaJCIS), IEEE, 2016, pp. 48–54.
[92] G. Aggarwal, N. Mishra, B. Pinkas, Secure computation of the median (and other elements of specified ranks), Journal of cryptology 23 (2010) 373–401.
[93] A. Shamir, How to share a secret, Commun. ACM 22 (1979) 612–613.
[94] T. Nishide, K. Ohta, Multiparty computation for interval, equality, and comparison without bit-decomposition protocol, in: PKC, 2007, pp. 343–360.
[95] J. Larson, S. Mattu, L. Kirchner, J. Angwin, How we analyzed the compas recidivism algorithm (May 2016).
URL https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm
[96] S. A. Friedler, C. Scheidegger, S. Venkatasubramanian, S. Choudhary, E. P. Hamilton, D. Roth, A comparative study of fairness-enhancing interventions in machine learning, in: Proceedings of the Conference on Fairness, Accountability, and Transparency, ACM, 2019, pp. 329–338.
[97] S. Moro, P. Cortez, P. Rita, A data-driven approach to predict the success of bank telemarketing, Decision Support Systems 62 (2014) 22–31.
[98] D. Dua, C. Graff, UCI machine learning repository (2017).
URL http://archive.ics.uci.edu/ml
[99] Y. Rubner, C. Tomasi, L. J. Guibas, A metric for distributions with applications to image databases, in: Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271), IEEE, 1998, pp. 59–66.
[100] I. Damgård, M. Geisler, M. Krøigaard, J. B. Nielsen, Asynchronous multiparty computation: Theory and implementation, in: International workshop on public key cryptography, Springer, 2009, pp. 160–179.

A SUMMARY OF NOTATIONS

- D - the horizontally distributed dataset.
- L - the number of parties.
- P_ℓ - a party, $\ell \in [L] \coloneqq \{1, \ldots, L\}$.
- W - a given population of individuals.
- A - a set of attributes that relate to each of the individuals.
- S - a sensitive attribute (e.g., race or gender). $S \in \{U, V\}$, where U means *unprivileged* and V means *privileged*.
- X - the non-sensitive attribute.
- Y - the binary class attribute that needs to be predicted (e.g. "hire/no hire").
- W^S - the subset of individuals in W that are associated with group S, $S \in \{U, V\}$.
- W_ℓ - the subset of individuals in W whose information is held by the party P_ℓ, $\ell \in [L]$.
- W_ℓ^S - $W^S \bigcap W_\ell$.
- n - $|W|$.
- n^S - $|W^S|$, for each $S \in \{U, V\}$.
- n_ℓ - $|W_\ell|$, for each $\ell \in [L]$.
- n_ℓ^S - $|W_\ell^S|$, for each $S \in \{U, V\}$, $\ell \in [L]$.
- $D(X)$ - the multiset of values appearing in the X-column of D.
- $D^S(X)$ - the multiset of values appearing in the X-column of D, restricted to the rows in W^S, for each $S \in \{U, V\}$.
- $D_\ell(X)$ - the multiset of values appearing in the X-column of D, restricted to the rows in W_ℓ, for each $\ell \in [L]$.
- $D_\ell^S(X)$ - the multiset of values appearing in the X-column of D, restricted to the rows in W_ℓ^S, for each $S \in \{U, V\}$, $\ell \in [L]$.
- B - the number of bins.
- $b_{(i)}^S$ - the i^{th} bin in a sorted quantile-based binning scheme, dividing $D^S(X)$ to nearly equal-sized bins.
- x - an original value of an individual.
- $\bar{x}$ - the repaired value of x.
- λ - the repair tuning parameter ($\lambda \in [0, 1]$).
- K^S - the list of locations of boundaries for all bins in $D^S(X)$, for each $S \in \{U, V\}$.
- q^S and r^S - the quotient and remainder, respectively, when dividing n^S by B, for each $S \in \{U, V\}$.
- $m_{(i)}^S$ - $\min\{b_{(i)}^S\}$, for each $S \in \{U, V\}$, $i \in [B]$.
- $m_{(B+1)}^S$ - $\max\{b_{(B)}^S\}$, for each $S \in \{U, V\}$.
- d - the precision of digits after the decimal point.
- $[\alpha, \beta]$ - the range of possible values in $D(X)$ (after they were multiplied by 10^d, for some d on which the parties agreed upfront, and rounded to the nearest integer).
- M - the number of possible values in $D(X)$, i.e., $\beta - \alpha + 1$.
- $\biguplus$ - a disjoint union.

B EXPERIMENTS WITH ADDITIONAL MACHINE LEARNING CLASSIFIERS

In Section 5.2 we presented experiments based on a Logistic Regression classifier. Here we present results with additional classifiers — Random Forest and Neural Network.

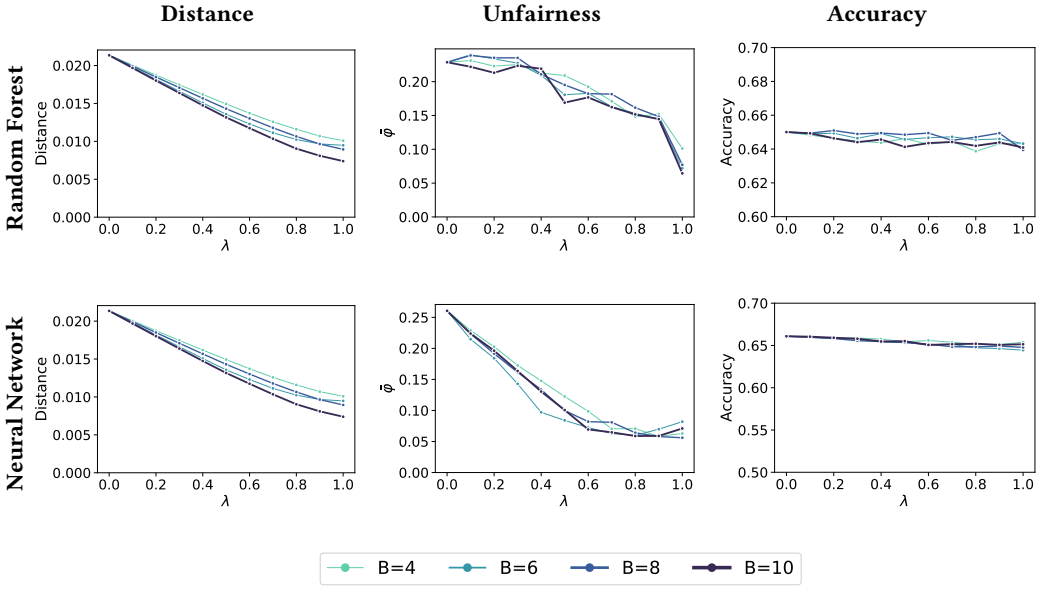

Fig. 10. The effect of the repair tuning parameter λ on distance, accuracy and unfairness (based on the ProPublica dataset). Increasing the value of λ yields a considerable decrease of distance and, consequently, also a decrease of unfairness, with only a minor compromise in accuracy.

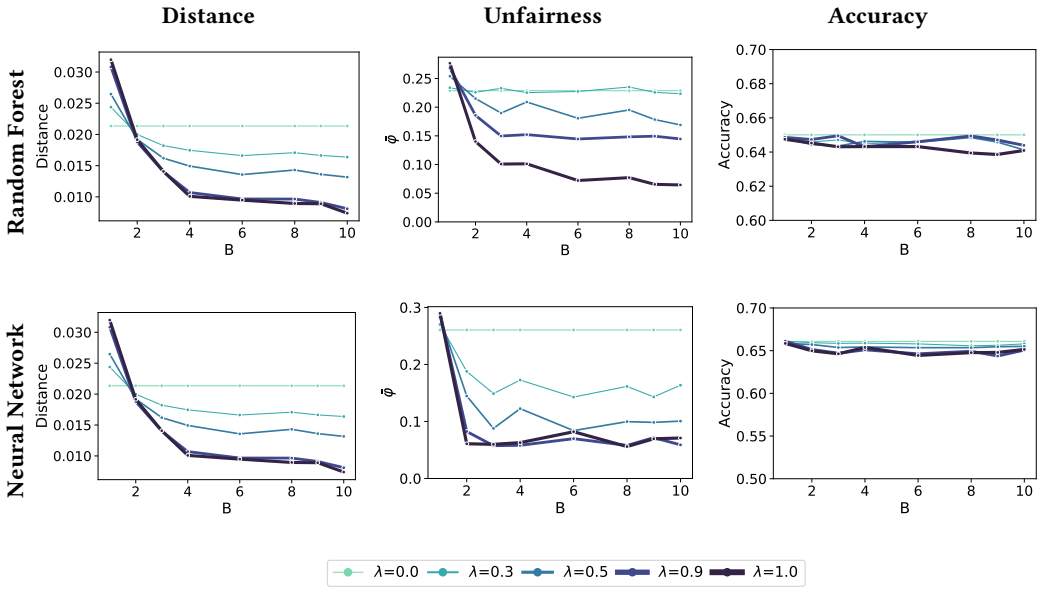

Fig. 11. The effect of the number of bins B on distance, accuracy and unfairness (based on the ProPublica dataset). Increasing the value of B leads to improvement in fairness with only a minor compromise in accuracy. However, increasing B has a diminishing marginal effect, namely, a large number of bins barely contributes to the decrease in unfairness.