Privacy-Preserving Federated Data Sharing

Ferdinando Fioretto
Georgia Institute of Technology
Atlanta, GA, USA
fioretto@gatech.edu

Pascal Van Hentenryck
Georgia Institute of Technology
Atlanta, GA, USA
pvh@isye.gatech.edu

ABSTRACT

Consider a set of agents with sensitive datasets who are interested in the same prediction task and would like to share their datasets without revealing private information. For instance, the agents may be medical centers with their own historical databases and the task may be the diagnosis of a rare form of a disease. This paper investigates whether sharing privacy-preserving versions of these datasets may improve the agent predictions. It proposes a Privacy-preserving Federated Data Sharing (PFDS) protocol that each agent can run locally to produce a privacy-preserving version of its original dataset. The PFDS protocol is evaluated on several standard prediction tasks and experimental results demonstrate the potential of sharing privacy-preserving datasets to produce accurate predictors.

ACM Reference Format:
Ferdinando Fioretto and Pascal Van Hentenryck. 2019. Privacy-Preserving Federated Data Sharing. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

We live in a world where a significant amount of information is collected by multiple entities on every one of us. This information fuels a variety of services from personalized product recommendations to personalized health-care. In general, the collected data contain sensitive information, e.g., regarding our health or our daily activities. As a result, entities collecting data may be reluctant to disclose their datasets, even though their release may have significant societal benefits. Consider for instance a university hospital developing a predictor for a rare disease. Such a predictor would likely benefit from augmenting the local dataset with those from other medical centers, especially if the local dataset is biased towards a sub-population. However, it has been shown that even when several data attributes have been omitted from the shared datasets, the predictor developed on the aggregated data is vulnerable to privacy attacks [9, 26].

Two different approaches have been considered in the literature to address this issue. In the first approach, an entity releases a privacy-preserving predictor as the output [4, 14, 18, 29] while, in the second approach, an entity releases a privacy-preserving synopsis of the dataset that can be used to train a predictor [6, 21, 27]. Publishing a privacy-preserving dataset is often preferred to publishing a privacy-preserving model, as it enables more general exploratory and predictive analytic tasks. However, most of the recent literature on privacy-preserving dataset focuses on releasing noisy histograms for low (one or two) dimensional datasets [5, 16, 28, 30], which are rarely employed to train learning models. Moreover, it was observed that these approaches perform poorly on high-dimensional data [25, 27]. In addition, these methods were designed for single entities and do not consider the multi-agent setting where several entities would like to share privacy-preserving versions of their datasets.

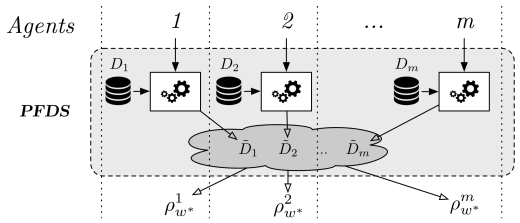

Figure 1: The PFDS framework.

This paper addresses this gap and proposes a Privacy-preserving Federated Data Sharing (PFDS) protocol that allows agents to privately build a high-fidelity dataset using the framework of *Differential Privacy* [7]. The protocol is sketched in Figure 1, which shows every agent releasing a privacy-preserving dataset. An agent can then use these datasets, together with its own, to train a variety of predictors. The PFDS protocol consists of two main steps that are executed locally by every agent. In the first step, an agent builds a locally trained privacy-preserving predictor that is shared with other agents. In a second step, the agent synthesizes an unlabeled privacy-preserving version of its data and uses the collected predictors to generate its labels, leveraging the knowledge transferred from all shared models. The agent then shares its privacy-preserving, labeled, data that can be used by other agents for their analytic tasks.

PFDS has two main benefits: (1) It allows multiple entities to train a predictor that leverage distributed data privately and (2) it allows agents to release a privacy-preserving dataset which can be used for a variety of tasks. The PFDS protocol was compared against state-of-the-art differential private ensembles of predictors for linear regression, logistic regression, and support vector machines. The experimental results indicate that PFDS reduces the error of competitor methods of up to 11% on several classification and regression tasks.

2 PROBLEM DEFINITION

This paper develops a privacy-preserving data sharing scheme for *regression* and *classification* tasks. This section reviews the single agent setting, presents the envisioned federated schema, and reviews the adopted privacy notion. $\|\mathbf{x}\|_1$ and $\|\mathbf{x}\|$ will denote the

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

l_1-norm and l_2-norm respectively, $[n]$ denotes the set $\{1, 2, \ldots, n\}$, boldface is used to denote vectors and calligraphic type for sets.

2.1 Empirical Risk Minimization

This paper considers a training dataset $D = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} : i \in [n]\}$ containing n labeled data points with data space $\mathcal{X} = \mathbb{R}^d$ and label set $\mathcal{Y} = [-1, 1] \subseteq \mathbb{R}$. A tuple $(x_{i1}, \ldots, x_{id}, y_i)$ in D describes the attributes of an individual (e.g., age, gender), encoded by values $\mathbf{x}_i$, and their relationship, encoded by label y_i. The support for the attribute $a \in [d]$ is denoted by Ω_a. Label y_i is real-valued for regression tasks and binary (e.g., -1 or 1) for classification tasks.

The objective is to construct a *predictor* $\rho_{\mathbf{w}} : \mathcal{X} \to \mathcal{Y}$, where $\mathbf{w}$ is a vector of real-valued parameters. For example, for linear regression, ρ is a linear combination of the $\mathbf{x}_i$ and $\mathbf{w}$ is a d-dimensional vector whose k-th element represents the weight of x_k. The quality of the predictor on the training data is measured via a nonnegative *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

Regularized empirical risk minimization (ERM) aims at choosing a predictor $\rho_{\mathbf{w}}$ that minimizes the regularized empirical loss:

$$J(\rho_{\mathbf{w}}, D) = \frac{1}{n} \sum_{i=1}^{n} \left(\ell\left(\rho_{\mathbf{w}}(\mathbf{x}_i), y_i\right)\right) + \lambda c(\mathbf{w}) \tag{1}$$

where the minimization is performed over the parameter space $\mathbf{w}$ of ρ, the regularizer $c(\cdot)$ is used to prevent over-fitting, and λ controls the degree of regularization. This paper focuses on three commonly ERM tasks: linear regression, logistic regression, and support vector machines (SVM). It also uses an l_2-regularizor: $c(\mathbf{w}) = \|\mathbf{w}\|^2$.

Linear Regression. A linear regression on a dataset D returns a predictor $\rho_{\mathbf{w}^*}$ minimizing the loss function:

$$\sum_{i=1}^{n} \left(\mathbf{w}^T \mathbf{x}_i - y_i\right)^2. \tag{2}$$

Logistic Regression. A logistic regression classifier on a dataset D returns a predictor $\rho_{\mathbf{w}^*}$ that predicts $y_i = 1$ with probability $\rho_{\mathbf{w}^*}(\mathbf{x}_i) = \frac{\exp(\mathbf{w}^{*T}\mathbf{x}_i)}{1+\exp(\mathbf{w}^{*T}\mathbf{x}_i)}$, where $\mathbf{w}^*$ is an n-dimensional real-valued vector minimizing the loss function:

$$\sum_{i=1}^{n} \left(\log\left(1 + \exp\left(-y_i(\mathbf{w}^T \mathbf{x}_i)\right)\right)\right). \tag{3}$$

For example, if $\mathbf{x}_i$ describes the gender, age, and blood type of a person, and y_i indicates whether the person has diabetes, then a logistic regression returns a function mapping her gender, age, and blood type to the probability that she has diabetes.

Support Vector Machines. A Support Vector Machine (SVM) classifier on a dataset D returns a predictor $\rho_{\mathbf{w}^*}$ where $\mathbf{w}^*$ minimizes the loss function:

$$\sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i)\right). \tag{4}$$

An SVM classifier finds the *maximum-margin hyperplane* that divides the group of points $\mathbf{x}_i$ for which $y_i = 1$ from those for which $y_i = -1$, by maximizing the distance between the hyperplane and the nearest point $\mathbf{x}_i$ from either group.

In this paper, *predictor* refers to an ERM algorithm.

2.2 The Multi-agent ERM

The multi-agent setting considers a set of m agents. Each agent $k \in [m]$ owns a dataset $D_k = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} : i \in [n_k]\}$ that contains n_k labeled data points. $\mathcal{D} = \cup_{k\in[m]} D_k$ is the union of all the agent datasets and this paper assumes that the datasets D_k ($k \in [m]$) are a partition of $\mathcal{D}$. Hence each entry in $\mathcal{D}$ describes an individual whose privacy must be protected.

The objective of each agent i is to train a predictor ρ_i. A standard option for an agent is to train the predictor on its own dataset D_i. However, the agent data may be biased toward a particular subpopulation and may not generalize to the population at large. One way to address this issue is to combine several models that have been trained on different datasets [2, 3, 15]. A simple, yet effective, scheme to aggregate multiple predictors is *majority voting*, which outputs the class that is predicted more often by predictors in the ensemble. Another widely adopted aggregation strategy is a *committee machine* [17], a meta predictor where multiple models are combined by averaging the results of each predictor. If the prediction errors made by the individual predictors are all uncorrelated and have 0 mean, the average error of a model could be reduced by a factor of m simply by averaging m members [23]. Even though typically errors are highly correlated, and the performance gain could be small, [24] shows that, even when committee members are correlated and biased, the squared prediction error of the committee (obtained through an averaging process) is no worse than the mean squared prediction error of the individual members.

Since agent data contains sensitive information, the shared models must be privacy-preserving. However, *a privacy-preserving model can be used solely to perform the model task.* This paper considers a more general setting in which a subset $\mathcal{S}$ of the n agent releases *a privacy-preserving version of their data* D_k that can be used together with the local data of agent i to train the predictor ρ_i. This methodology has the advantage of allowing multiple analytic tasks on the privacy-preserving data.

2.3 Differential Privacy

To protect the privacy of individuals in the datasets, this paper adopts the framework of *Differential Privacy* [8], which has emerged as the de-facto standard for privacy protection. A randomized algorithm $\mathcal{A} : \mathscr{D} \to \mathscr{R}$ with domain $\mathscr{D}$ and range $\mathscr{R}$ is ϵ-differential private if, for any output response $O \subseteq \mathscr{R}$ and any two *neighboring* inputs $D_1, D_2 \in \mathscr{D}$ differing in at most one individual (written $D_1 \sim D_2$),

$$Pr[\mathcal{A}(D_1) \in O] \leq \exp(\epsilon)\, Pr[\mathcal{A}(D_2) \in O], \tag{5}$$

where the probability is calculated over the coin tosses of $\mathcal{A}$. The parameter ϵ is the *privacy budget* of the algorithm, with values close to 0 denoting strong privacy. The budget is commonly set to values no greater than 1. Differential privacy satisfies several important properties, including *composability* and *immunity to post-processing*.

Composability ensures that a combination of differential private algorithms preserve differential privacy [8].

Theorem 2.1 (Sequential Composition). *The composition* $(\mathcal{A}_1(D), \ldots, \mathcal{A}_k(D))$ *of a collection* $\{\mathcal{A}_i\}_{i=1}^k$ *of* ϵ_i*-differential private algorithms satisfies* $(\sum_{i=1}^k \epsilon_i)$*-differential privacy.*

Theorem 2.2 (Parallel Composition). *Let D_1 and D_2 be disjoint subsets of D and $\mathcal{A}$ be an ϵ-differential private algorithm. Computing $\mathcal{A}(D \cap D_1)$ and $\mathcal{A}(D \cap D_2)$ satisfies ϵ-differential privacy.*

Post-processing immunity ensures that privacy guarantees are preserved by arbitrary post-processing steps [8].

Theorem 2.3 (Post-Processing Immunity). *Let $\mathcal{A}$ be an ϵ-differential private algorithm and g be an arbitrary mapping from the set of possible output sequences O to an arbitrary set. Then, $g \circ \mathcal{A}$ is ϵ-differential private.*

The following differential private algorithms are building blocks.

2.3.1 The Laplace Mechanism. In private data analysis settings, agents interact with the dataset by issuing queries. A (numeric) *query* is a function from a data set $D \in \mathscr{D}$ to a result set $R \subseteq \mathbb{R}^d$. A query Q can be made differential private by injecting random noise to its output. The amount of noise to inject depends on the *sensitivity* of the query, denoted by Δ_Q and defined as

$$\Delta_Q = \max_{D_1 \sim D_2} \|Q(D_1) - Q(D_2)\|_1 .$$

In other words, the sensitivity of a query is the maximum l_1-distance between the query outputs from any two neighboring dataset D_1 and D_2. For instance, $\Delta_Q = 1$ for a query Q that counts the number of users satisfying a given property in a dataset.

The Laplace distribution with 0 mean and scale b, denoted by $\text{Lap}(b)$, has a probability density function $\text{Lap}(x|b) = \frac{1}{2b} e^{-\frac{|x|}{b}}$. It can be used to obtain an ϵ-differential private algorithm to answer numeric queries [7]. In the following, $\text{Lap}(\lambda)^d$ denotes the i.i.d. Laplace distribution over d dimensions with parameter λ.

Theorem 2.4 (Laplace Mechanism $\mathcal{M}_{\text{Lap}}$). *Let Q be a numeric query that maps datasets to $\mathbb{R}^d$. The Laplace mechanism that outputs $Q(D) + z$, where $z \in \mathbb{R}^d$ is drawn from the Laplace distribution $Lap\left(\frac{\Delta_Q}{\epsilon}\right)^d$, achieves ϵ-differential privacy.*

2.3.2 The Exponential Mechanism. While the Laplace mechanism allows to answer numeric queries privately, it cannot be applied to queries with non-numeric value outputs. The *Exponential Mechanism* [20] overcomes this limitation by releasing a private answer to a query by sampling from its discrete output space O. The sampling probability for each output $o \in O$ is based on a user-specified *utility function* $u : (\mathscr{D} \times O) \to \mathbb{R}$ that assigns a real value score to each output o for a given input dataset. Higher scores denote more desirable outputs and the likelihood of a selection for an output grows exponentially with its score value.

Theorem 2.5 (Exponential Mechanism M_{EXP}). *Let $u : (\mathscr{D} \times O) \to \mathbb{R}$ have sensitivity $\Delta_u = \max_{o \in O} \max_{D_1 \sim D_2} |u(D_1, o) - u(D_2, o)|$. The exponential mechanism that outputs o such that $\Pr[o \text{ is selected}] \propto \exp\left(\frac{\epsilon u(D,o)}{2\Delta_u}\right)$ satisfies $\epsilon-$differential privacy.*

3 THE PFDS FRAMEWORK

This paper introduces a simple, yet effective, framework for training an agent predictor taking advantage of privacy-preserving data shared by multiple agents. The Privacy-preserving Federated Data Sharing (PFDS) framework for ERM is illustrated in Figure 2 and

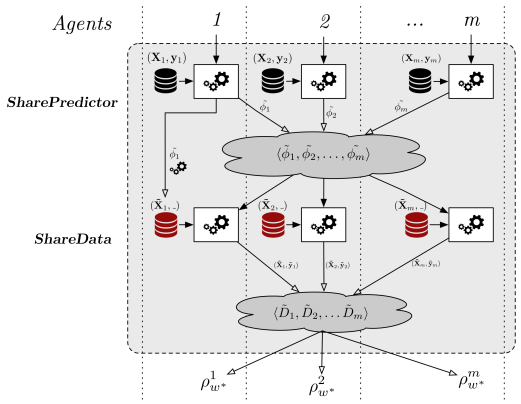

Figure 2: Detailed illustration of the PFDS framework.

Algorithm 1: *PFDS*(D_i, H, ϵ)

1 *SharePredictor*(D_i, H, ϵ);
2 *ShareData*$(\{\tilde{\phi}_j\}_{j \in \mathcal{S}})$;
3 $w* \leftarrow \text{argmin}_w J(\rho_w, \cup_{j \in \mathcal{S}} \tilde{D}_j \cup D_i)$;
4 **return** ρ^i_{w*}

described in Algorithm 1, which is executed by each agent. The subset of agents sharing their model and data privately (the *sharers*) is denoted by $\mathcal{S}$.

The algorithm takes as input the agent data D_i, a set of hyper-parameters H (to be defined later), and a privacy budget $\epsilon > 0$. PFDS is composed of two main phases: The *Predictor Sharing* phase and the *Data Sharing* phase. For every sharer $i \in \mathcal{S}$, the *Predictor Sharing* phase generates and shares a privacy-preserving predictor $\tilde{\phi}_i$ trained on data D_i. The sharer runs this algorithm locally. Note that $\tilde{\phi}_i$ can also be used to construct a data synopsis for D_i. The output of the *Predictor Sharing* phase is thus the collection of all shared privacy-preserving predictors $\{\tilde{\phi}_j\}_{j \in \mathcal{S}}$ (line 1). The *Data Sharing* phase generates and shares a privacy-preserving dataset $\tilde{D}_i$, by leveraging the shared predictors (line 2). Once again, it is run locally by each agent. Collectively, these two phases allow each agent to receive a collection of privacy-preserving dataset $\{\tilde{D}_j\}_{j \in \mathcal{S}}$ that, together with the agent data D_i, can be used for various analytic tasks. In this paper, the task amounts to minimizing an empirical loss function, defined in Equation (1), in order to produce a predictor $\rho^i_{w^*}$ (line 3) that can be used by the agent.

The rest of this section describes the *Predictor Sharing* and *Data Sharing* phases in detail. $\mathbf{X}_i$ and $\tilde{\mathbf{X}}_i$ denote the tensor of tuples that excludes the labels from D_i and its privacy-preserving version, respectively. Similarly, $\mathbf{y}_i$ and $\tilde{\mathbf{y}}_i$ denote the vector of labels associated with the values in D_i and its privacy-preserving version, respectively.

3.1 The *Predictor Sharing* Phase

The *Predictor Sharing* phase (run locally by each agent) is described in Algorithm 2 and relies on building a type of differential private decision tree. The algorithm takes, as inputs, the agent data D_i, the privacy budget ϵ, and a list of hyper-parameters $h_{\max}$, t, and p.

Algorithm 2: *SharePredictor* $(D_i, H = \langle h_{\max}, t, p \rangle, \epsilon)$

1 **if** *agent* $i \in \mathcal{S}$ **then**
2 $\quad \tilde{\phi}_i \leftarrow$ *makeTree* $(D_i, \emptyset, A, 1, h_{\max}, t, \frac{\epsilon}{2h_{\max}})$;
3 $\quad$ **send** $\tilde{\phi}_i$ to all agents $j \in \mathcal{S} \setminus \{i\}$;
4 **receive** $\tilde{\phi}_j$ from all agents $j \in \mathcal{S} \setminus \{i\}$;

Each sharer constructs a differential private decision tree $\tilde{\phi}_i$ fitting its data D_i (line 2). It then sends its predictor $\tilde{\phi}_i$ to every other agent in $\mathcal{S}$ (line 3). In Figure 2, this step is illustrated by depicting the agents taking as input their data and sharing the resulting privacy-preserving models into the cloud. Finally, the agents wait to receive all shared predictors (line 4). This model ensemble will be used in the next phase to find good labels for the privacy-preserving unlabeled data built by the agent.

The PFDS protocol first builds a privacy-preserving predictor $\tilde{\phi}_i$ from dataset D_i. The predictor will be shared with other agents and used to create an unlabeled privacy-preserving version of $\mathbf{X}_i$. More precisely, $\tilde{\phi}_i$ is used to find a partition of $\mathcal{X}$ and create a data synopsis for D_i. Once a data partition has been created, one can construct a *histogram* of the data by merely counting the number of individuals for each class in each set of the partition. In this paper, each agent grows a decision tree whose paths from the root to each leaf is used to derive a partition of their data.

The following notations are used. Let A be the set of data attributes of the dataset D. Given an attribute $a \in A$, a dataset D, and a value $v \in \Omega_a$, $D_{a=v}$ refers to the projection of D that includes only the tuples in which attribute a takes on value v, i.e., $D_{a=v} := \{(\mathbf{x}_i, y_i) \in D \mid x_{ia} = v\}$. Similarly, $D_{a<v}$ and $D_{a\geq v}$ refer to the projections of D whose tuples satisfy the relations $a < v$ and $a \geq v$, respectively. For a label $l \in \mathcal{Y}$, $D^{y=l} := \{(\mathbf{x}_i, y_i) \in D | y_i = l\}$. Given a tree structure T, T_{ij} denotes the node at level i (with 1 denoting the root level), and position j in the sibling set of level i for a given sibling order $\prec_S$. The notation T_{ij}.ch describes the set of children nodes of node T_{ij}, T_{ij}.data denotes the subset of data held at node T_{ij} and, for a leaf node T_{ij}, T_{ij}.label denotes the outcome of the decision path from the root node to T_{ij}. Finally, m_i denotes the number of nodes in T at level i. These elements are constructed implicitly during the tree construction.

The *makeTree* process is described in Algorithm 3, which is executed locally by each sharer $i \in \mathcal{S}$. The algorithm takes as input the agent dataset (called D, in lieu of numeric subscripts), the tree T being grown, the set of attributes A that can be used during the construction of a subtree, the current tree level h being explored, the maximum depth of the tree $h_{\max} \leq |A|$, an integer t used during value selections of attributes, and the privacy budget ϵ. For non-leaf nodes (lines 1–11), the algorithm saves the current attribute set (line 2) and selects a new attribute a (line 3). The *selectAttribute* function samples at random one attribute a from the set A and, if a happens to be categorical, the agent updates the set A by removing a from it (lines 17–20). If the selected attribute a is numerical, a split value is selected (line 5), and Algorithm 3 is called recursively on each side of the split (lines 6–7). The *selectSplit* function for numerical attributes is described in lines (21–23). It first selects at random a set V of t values from the domain of the attribute a that is active in the current tree node (line 21). It then uses the exponential mechanism to select a value v from V using the following utility function:

$$u(D, a, v) = \max_{l \in \mathcal{Y}} \left| D^{y=l}_{a<v} \right| + \max_{l \in \mathcal{Y}} \left| D^{y=l}_{a\geq v} \right|.$$

The above is referred to as *Max operator* and was first proposed in [14]. The Max operator has sensitivity $\Delta_u = 1$ [14]. While different utility functions could be used (e.g., Gini index or information gain), the Max operator was shown to outperform other metrics in private tree classifications [14, 21]. For a task where the class attributes are continuous, the algorithm discretizes $\mathcal{Y}$ into $b = 10$ buckets and maps each data element label to one of these buckets.

When the selected attribute a is categorical, Algorithm 3 branches the current tree for each element in the domain of a (lines 9–10). In such a case, the attribute set A is restored on returning from the recursive call. Finally, when a leaf node is reached (i.e., when the current level h reaches $h_{\max}$), the process uses the Laplace mechanism to add noise on the elements count in each class for the data associated with the leaf node (line 13). These counts are then stored to estimate the data size at the node (line 14) and to select the label associated with the node (line 15). The label is chosen by selecting the class associated with the highest noisy count.

Algorithm 3: *makeTree* $(D, T, A, h, h_{max}, t, \epsilon)$

1 **if** $h < h_{\max}$ **then**
2 $\quad A' \leftarrow A$;
3 $\quad (a, A) \leftarrow$ *selectAttribute* (A);
4 $\quad$ **if** *a is numerical* **then**
5 $\quad\quad v \leftarrow$ *selectSplit* (D, a, ϵ);
6 $\quad\quad T \leftarrow T \cup$ *makeTree* $(D_{a<v}, T, A, h+1, h_{\max}, t, \epsilon)$;
7 $\quad\quad T \leftarrow T \cup$ *makeTree* $(D_{a\geq v}, T, A, h+1, h_{\max}, t, \epsilon)$;
8 $\quad$ **else**
9 $\quad\quad$ **foreach** $v \in \Omega_a$ **do**
10 $\quad\quad\quad T \leftarrow T \cup$ *makeTree* $(D_{a=v}, T, A, h+1, h_{\max}, t, \epsilon)$;
11 $\quad\quad A \leftarrow A'$;
12 **else**
13 $\quad c_l = |D^{y=l}| + \text{Lap}(1/\epsilon) \;\; (l \in \mathcal{Y})$;
14 $\quad T \leftarrow T \cup \{(l, c_l) \mid l \in \mathcal{Y}\}$;
15 $\quad T$.label $\leftarrow \text{argmax}_{l \in \mathcal{Y}} c_l$;
16 **return** T

Function selectAttribute(A)**:**
17 $\quad a \leftarrow$ sample from $U(A)$;
18 $\quad$ **if** *a is categorical* **then**
19 $\quad\quad A \leftarrow A \setminus \{a\}$
20 $\quad$ **return** *(a, A)*

Function selectSplit(a, D, ϵ')**:**
21 $\quad V \leftarrow$ sample t values from active Ω_a;
22 $\quad$ Select $v \in V$ with probability $\propto \exp\left(\frac{\epsilon}{2\Delta_u} u(D, a, v)\right)$
23 $\quad$ **return** v

Theorem 3.1. *Algorithm 3 is* $h_{\max}\epsilon$*-differential private.*

Proof. The algorithm queries the dataset to (i) select a split value (in line 5) and to (ii) count the number of occurrences at the

Algorithm 4: *ShareData* $(\{\tilde{\phi}_j\}_{j\in\mathcal{S}})$

1 **if** *agent* $i \in \mathcal{S}$ **then**
2 $\quad \tilde{\mathbf{X}}_i \leftarrow$ *optPrivateGen* $(\tilde{\phi}_i, D_i, p, \frac{\epsilon}{2})$;
3 $\quad \tilde{D}_i \leftarrow \emptyset$;
4 $\quad$ **foreach** $\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}_i$ **do**
5 $\quad\quad L \leftarrow \{\tilde{\phi}_j(\tilde{\mathbf{x}}_i) \ : \ j \in \mathcal{S}\}$;
6 $\quad\quad \tilde{y}_i \leftarrow$ majorityVote(L);
7 $\quad\quad \tilde{D}_i \leftarrow \tilde{D}_i \cup (\tilde{\mathbf{x}}_i, \tilde{y}_i)$
8 $\quad$ **send** $\tilde{D}_i$ to all agents $j \in [n]$;
9 **receive** $\tilde{D}_j$ from all agents $j \in \mathcal{S}$;

leaf nodes (line 13). A split value is selected using the exponential mechanism, that considers t randomly selected values and has parameter $\frac{\epsilon}{2\Delta_u}$. It thus satisfies ϵ-differential privacy by Theorem 2.5. The maximum length of a path from the root to a leaf node is $h_{max} - 1$, thus the split selection process ensures $(h_{\max} - 1)$-differential privacy for every path of the tree by sequential composition (Theorem 2.1). For a given level h of the tree, the data of the sibling $T_{hi}.data$ $(i \in [m_h])$ forms a partition for D. Thus, by parallel composition (Theorem 2.2), the split selection process ensures $(h_{\max} - 1)$-differential privacy for each path of the tree from the root node to a leaf node. Finally, for each leaf node T_{hi}, the algorithm queries the size of the dataset $T_{hi}.data$ for each class in $\mathcal{Y}$. Notice that an element in $T_{hi}.data$ can be labeled in exactly one class in $\mathcal{Y}$. Additionally, the sensitivity of a count query is one, and thus, by sequential and parallel composition, the overall privacy guarantee of Algorithm 3 is $h_{\max}\epsilon$. □

3.2 The *Data Sharing* Phase

After the *Predictor Sharing* phase, each sharer i has access to the set of privacy-preserving predictors $\{\tilde{\phi}_j\}_{j\in\mathcal{S}}$ and is ready to perform the *Data Sharing* phase, described in Algorithm 4. The agent starts by creating an unlabeled privacy-preserving version $\tilde{\mathbf{X}}_i$ of its dataset D_i and she then uses the set of privacy-preserving predictors to obtain the labels of $\tilde{\mathbf{X}}_i$. The resulting data set is then shared with everyone. The rest of this subsection presents the details of these operations.

3.2.1 ***Optimization-based Private Data Construction.*** To obtain a privacy-preserving version of her own dataset, sharer i uses the optimization-based framework introduced in [13]. Indeed, the noise introduced at the leaves by Algorithm 3 may substantially alter the size of each partition of the tree, so that some global properties of interest, such as the total number of individuals in the dataset or the number of individuals having a given property, may differ from their original values. Hence, *optPrivateGen* function starts by redistributing this noise before generating the dataset.

This process is described in Algorithm 5. It computes the final estimates $\hat{\mathbf{x}}$ for the partition sizes induced by T, using the noisy sizes $\tilde{\mathbf{x}}$ of each set in the partition extracted from T and additional queries over D. The idea is to use the information encoded in the higher branches of the tree to compute the statistics to retain in the final data release. The algorithm takes as input the private tree T, which includes the noisy partition sizes at the leaf level, the dataset D, a

Algorithm 5: *optPrivateGen* (T, D, p, ϵ)

1 $\mathbf{c} = (|T_{ij}.\text{data}| \ \text{ for all } i = 1, \ldots, p-1, h_T; j = 1 \ldots, m_i)$;
2 $\tilde{\mathbf{c}} = \mathcal{M}_{\text{Lap}}(\mathbf{c}, \ \frac{\epsilon}{p-1})$;
3 $\mathbf{x}^* = \mathbf{argmin}_{\dot{\mathbf{x}}} \|\dot{\mathbf{x}} - \tilde{\mathbf{c}}\|^2_{2,\lambda} =$

$$\sum_{i=1}^{p} \frac{1}{m_i} \sum_{j=1}^{m_i} (\dot{x}_{ij} - \tilde{c}_{ij})^2 \qquad \text{(O1)}$$

$$\textbf{subject to} : \forall i \in [p-1], j \in [m_i] : \ \dot{x}_{ij} = \sum_{l \in T_{ij}.\text{ch}} \dot{x}_{i+1\,l} \qquad \text{(O2)}$$

$$\forall i, j : \ \dot{x}_{ij} \geq 0. \qquad \text{(O3)}$$

4 $\hat{\mathbf{x}} = x^*_{p1}, \ldots, x^*_{p\,m_p}$;
5 $\tilde{\mathbf{X}} \leftarrow \emptyset$;
6 **foreach** $j \in [m_p]$ **do**
7 $\quad \tilde{\mathbf{X}} \leftarrow \tilde{\mathbf{X}} \cup \{x^*_{ij}$ individuals with attributes and labels sampled uniformly at random in path T_{11} to $T_{pj}\}$
8 **return** $\tilde{\mathbf{X}}$

value p denoting the number of levels to analyze, and the privacy budget ϵ. It first queries the size of each dataset associated with the tree nodes up to level $p-1$, as well as all the leaves (whose levels are denoted to as h_T). Such information is stored into a vector of counts $\mathbf{c}$ (line 1). These count queries are made differential private by applying the Laplace mechanism with parameter $\frac{\epsilon}{p-1}$ (line 2) to each query associated with tree levels $1 \ldots p-1$ (excluding the counts at the leaf level that are already privacy-preserving). The result is a new vector of counts $\tilde{\mathbf{c}}$. For notational simplicity, index p denotes level h_T and, for a node $T_{p-1\,j}$ at level $p-1$, $T_{p-1\,j}.ch$ denotes the leaves of T whose subtrees are rooted at $T_{p-1\,j}$. Similarly, $x_{p,j}$ denotes $x_{h_T\,j}$ and m_p denotes m_{h_T}.

The resulting values $\tilde{\mathbf{c}} = (\tilde{c}_{11}, \ldots, \tilde{c}_{p\,m_p})$ are then post-processed by the optimization algorithm depicted in line (3) to obtain the values $\mathbf{x}^* = (x^*_{11}, \ldots, x^*_{p\,m_p})$. Next, the algorithm extracts the vector $\hat{\mathbf{x}} = (x^*_{p1}, \ldots, x^*_{p\,m_p})$ (line 4) representing the final size of the partitions induced by T. Finally. the algorithm synthesizers a new (unlabeled) dataset $\langle \tilde{\mathbf{X}}, \_ \rangle$ by generating x^*_j individuals, for each $j \in [m_p]$, with values sampled in the intervals defined by the tree T following the path from its root node to its leaf T_{pj} (lines 5–7).

The essence of Algorithm 5 is the optimization model depicted in line (3). Its decision variables are the post-processed values $\dot{\mathbf{x}} = (\dot{x}_{11}, \ldots, \dot{x}_{pm_p})$, and $\lambda = (\lambda_1, \ldots, \lambda_p) \in (0, 1]^p$ is a vector of reals representing weights for the terms of the objective function. The objective minimizes the squared weighted l_2-norm of $\dot{\mathbf{x}} - \tilde{\mathbf{c}}$, where the weight λ_i of element $x_{ij} - \tilde{c}_{ij}$ is $\frac{1}{m_i}$.

The optimization is subject to a set of *consistency constraints* among partitions sizes in each consecutive level of the tree T. For each level $i \in [p-1]$ and index $j \in [m_i]$, constraint O2 selects an element x_{ij} associated with node T_{ij} and all its children and imposes the constraint $\dot{x}_{ij} = \sum_{l \in T_{ij}.ch} \dot{x}_{i+1\,l}$, which ensures that the post-processed value $\dot{x}_{ij}$ is consistent with the sum of the post-processed values of its partition in $T_{ij}.ch$. By tree construction, the union of the set of nodes in $T_{ij}.ch$ is equal to T_{ij}.

Theorem 3.2. *The optimization-based post-process achieves ϵ-differential privacy.*

Proof. For level i, the set of nodes T_{ij} of T ($j \in m_i$) partitions the dataset. Since each query on the data is a count query, whose sensitivity is 1, then each $\tilde{c}_{ij}$ ($i < p$) obtained from the Laplace mechanism is $\frac{\epsilon}{p-1}$-differential-private by Theorem 2.4. The values $\tilde{c}_{p1}, \ldots, \tilde{c}_{p\,m_p}$ are differential private since the tree leaf counts are private (Theorem 3.1). Therefore, $(\tilde{c}_{11}, \ldots, \tilde{c}_{p\,m_p})$ is ϵ-differential-private by sequential composition (Theorem 2.1). Finally, the result follows from post-processing immunity (Theorem 2.3). □

The optimization model is convex and can be solved efficiently.

3.2.2 ***Labeling the Private Data Construction.*** To label $\tilde{\mathbf{X}}_i$, each agent i locally executes the following steps. For each entry $\tilde{\mathbf{x}}_i$, the agent assembles an ensemble L of predictions using each predictor $\tilde{\phi}_j$, for $j \in \mathcal{S}$, to predict a label for the entry $\tilde{\mathbf{x}}_i$ (line 4). Next, it selects the label $\tilde{y}_i$ that appears with the highest frequency in L (line 5) and stores the labeled data entry $(\tilde{\mathbf{x}}_i, \tilde{y}_i)$ into the dataset $\tilde{D}_i$ (line 6). Recall that, for a task where the class attributes are continuous, PFDS discretizes $\mathcal{Y}$ into $b = 10$ buckets and maps labels of each data element to one of these buckets. The result of the majority vote returns a bucket descriptor as a label. To generate a label $\tilde{y}_i$ in the original space $\mathcal{Y}$, PFDS samples a value from the uniform distribution whose support is the values encoded by the selected bucket. Next, the agent sends its privacy-preserving data $\tilde{D}_i$ to all other sharers. Note that this step does not violates privacy, since Algorithm 3 only accesses the predictor ensemble $\{\tilde{\phi}_j\}_{j\in\mathcal{S}}$ and the dataset $\tilde{\mathbf{X}}_i$ that are already differential private. Finally, all agents $i \in [m]$ collect the privacy-preserving version of the dataset shared by all sharers (line 8) which can be used by the agent for a data analysis task.

In their final step of Algorithm 1, each agent produces a predictor for an ERM task. This step is described in line (3) of Algorithm 1. To do so, each agent $i \in [m]$ uses the privacy-preserving datasets $\tilde{D}_j$ from all sharer agents $j \in \mathcal{S}$ together with its own dataset D_i. The resulting data is used to solve the associated ERM problem. This step does not incur any further privacy cost to an agent. Indeed, each agent uses a collection of differential private datasets received from the sharing agents as well as her dataset. Recall that the agent predictors are not released but used internally by each agent, and thus no additional privacy cost needs to be considered.

Theorem 3.3. *PFDS is ϵ-differential private.*

Proof. The result extends directly from Theorems 3.1 and 3.2 and hence the privacy budgets allocated for Algorithms 3 and 5 are $\epsilon/2h_{\max}$ and $\epsilon/2$ respectively. The result follows by Theorems 3.1 and 3.2 and composition of differential privacy (Theorem 2.1). □

4 EXPERIMENTAL ANALYSIS

This section evaluates the effectiveness of PFDS in producing synthetic datasets with high fidelity, its efficiency, and its prediction quality on three ERM tasks: linear regression, logistic regression, and support vector machines.

Datasets and Experimental Setting. Each experiment executes a ten-fold stratified cross-validation to evaluate the misclassification rate or the mean squared error of the predictors. equally sized subsets, preserving the percentage of samples for each class for classification tasks. The training data is partitioned among the agents as follows. For a fixed numerical attribute a, each agent i is assigned a value a_i sampled uniformly at random from Ω_a. Then, each training sample $(\mathbf{x}, y)$ is assigned to agent i with probability inversely proportional to the distance $|\mathbf{x}_a - a_i|$. For each train-test pair, the agents use their training data to privately compute a predictor or privately publish a dataset and build predictor on it. The predictor accuracy is evaluated on the test data, which is disjoint from the train data. The experiments report the average results over ten runs and ten-fold cross-validations. The analysis varies the privacy budget ϵ from 0.1 to 1.0 and the number of sharers participating in the problem.

The experiments are executed on 5 real datasets summarized in Table 1. In addition to the dataset name, the table reports its number of entries (n), the number of numerical (d_{num}) and categorical (d_{cat}) attributes, the task for which the dataset has been used (classification (cl) or regression (re)), and the number p of agents used in each experiment associated with the corresponding dataset.

Mechanisms. PFDS is compared against two state-of-the-art methods that output a differential private predictor: the *objective perturbation* [4], used for logistic regression and SVM tasks, and the *functional mechanism* [29], used for linear regression. They are described in detail in Section 5. These methods use the parameters suggested in the corresponding papers, and their multi-agent aggregation processes are executed using a majority vote scheme, for classification tasks, and a committee aggregation scheme for regression tasks, as described in Section 2.2. In the experiments, they are both denoted as *DP-Pred* whose meaning is clear from the nature of the task. Additionally, PFDS is compared against *DiffGen* [21], a method to publish differential private histograms for classification. DiffGen takes as input a dataset and a taxonomy tree, that describes the taxonomy of each data attribute. The algorithm constructs a partition of the dataset by exploring the given taxonomy tree. Similar to the construction of a (privacy-preserving) decision tree, each step selects an attribute for specialization using the exponential mechanism with the max operator as a scoring function. Once a tree is constructed, a noisy histogram is derived from its leaves and a synthetic dataset is generated from such histogram. The privacy budget is evenly distributed among all the specialization steps and the perturbation step executed to generate a noisy histogram, resulting in a $\epsilon/2(d + 2h_{max})$ budget per query.

All the mechanisms are implemented in Python 3 and Gurobi 7.0. PFDS and DiffGen use a standard implementation of the predictors, executed using the privately synthesized datasets. When not otherwise specified, PFDS and DiffGen generate trees of maximum depth $h_{\max} = 8$. Additionally. PFDS uses a number of samples $t = 10$ (see Algorithm 3) and a number of levels $p = 4$ for the optimization process associated with the data release (Algorithm 5).

Data Fidelity. The first results focus on PFDS ability to produce synthetic datasets of high fidelity. In addition to DiffGen, we compare PFDS against a version that disables the optimization step (line (3) of Algorithm 5) and is referred to as PFDS$^-$. The experiments evaluate the algorithms on the Census dataset, for which an attribute taxonomy for DiffGen is available and described in [21],

dataset	n	d_{num}	d_{cat}	task	p
Census Income [19]	48842	8	6	cl	100
Default of credit card [22]	45211	13	10	cl	100
Diabetes [1]	1151	19	0	cl	10
Bike sharing [10]	17389	8	8	re	100
Online News Popularity [11]	39797	47	14	re	100

Table 1: Datasets

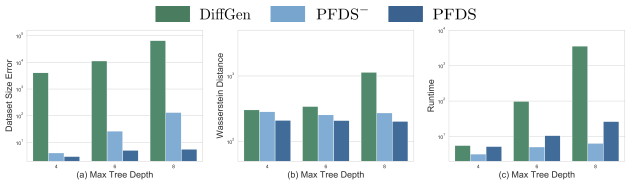

Figure 3: Comparison of the Data Synthesis Methods.

and use a privacy budget $\epsilon = 1.0$. *DiffGen cannot be evaluated on the remaining datasets for which no taxonomy is available.*

Figure 3(a) illustrates the average error of the dataset sizes, in log-10 scale, at varying of the maximum tree depth $h_{\max}$. The error is defined as the distance between the original and privacy-preserving dataset sizes. The figure clearly shows that PFDS synthesizes datasets whose sizes are closer to the original ones. In particular, the effect of the optimization step provides an improvement of up to one and three orders of magnitudes when compared to PFDS$^-$ and DiffGen, respectively. Data partitioning methods tend to overestimate the overall dataset size, especially in highly sparse settings, as in the current case [12, 13, 16]. DiffGen exasperates this behavior due to the higher amount of noise used to estimate the partition sizes. Finally, PFDS with the optimization step is able to retain higher data fidelity structure, thanks to the process of by enforcing consistency constraints on counts.

Figure 3(b) provides further insight into the data fidelity. It represents the average distance between the histograms obtained by randomly partitioning the data universe on the original data vs. the synthetic ones. The metric used is the *Wasserstein distance* and, for each experiment, the above process is repeated 50 times. Intuitively, the histograms induced from high-fidelity synthetic datasets will be close to those induced from the original data, regardless of the partitioning scheme. Once again, the results illustrate that PFDS is able to better capture the original data distribution.

The experiments also assess the runtime of the algorithms. Figure 3(c) reports the average time taken by the agents to generate their datasets. The results, shown in log-10 scale, illustrate that PFDS is at least one order of magnitude faster than DiffGen, for $h_{\max} > 4$, and that the overhead of the optimization step is relatively small.

Predictor Accuracy The quality of predictions of PFDS on all datasets is now evaluated and compared it against the appropriate differential private predictors (denoted by DP-pred as mentioned earlier), as well as against a non-private predictor (NP-Agt) executed by each agent on its own (non-private) dataset, which serves as a baseline. Finally, PFDS is also compared against a version that does not use shared predictors from other agents in order to label its data (i.e., it skips line 1 of Algorithm 1 and uses its predictor only as input to the *ShareData* algorithm), called PFDS(s). This comparison is provided to show the benefits of leveraging the knowledge transferred from all shared privacy-preserving models.

			Error			
dataset	ERM task	ϵ	NP-Agt	DP-Pred	PFDS(s)	PFDS
Census Income	Log. Reg.	1.0	0.276	0.232	0.266	0.212
Census Income	Log. Reg.	0.5	0.276	0.245	0.301	0.226
Census Income	Log. Reg.	0.1	0.276	0.293	0.336	0.246
Census Income	SVM	1.0	0.283	0.253	0.267	0.209
Census Income	SVM	0.5	0.283	0.262	0.269	0.224
Census Income	SVM	0.1	0.283	0.269	0.268	0.239
Credit	Log. Reg.	1.0	0.365	0.339	0.321	0.277
Credit	Log. Reg.	0.5	0.365	0.352	0.392	0.294
Credit	Log. Reg.	0.1	0.365	0.418	0.499	0.301
Credit	SVM	1.0	0.351	0.347	0.358	0.289
Credit	SVM	0.5	0.351	0.366	0.404	0.308
Credit	SVM	0.1	0.351	0.379	0.409	0.310
Diabetes	Log. Reg.	1.0	0.396	0.334	0.380	0.279
Diabetes	Log. Reg.	0.5	0.396	0.349	0.411	0.294
Diabetes	Log. Reg.	0.1	0.396	0.389	0.434	0.345
Diabetes	SVM	1.0	0.423	0.441	0.397	0.311
Diabetes	SVM	0.5	0.423	0.449	0.431	0.322
Diabetes	SVM	0.1	0.423	0.456	0.434	0.346
Bike sharing	Lin. Reg.	1.0	0.377	0.270	0.206	0.127
Bike sharing	Lin. Reg.	0.5	0.377	0.281	0.225	0.154
Bike sharing	Lin. Reg.	0.1	0.377	0.296	0.220	0.157
News popularity	Lin. Reg.	1.0	0.194	0.176	0.121	0.079
News popularity	Lin. Reg.	0.5	0.194	0.178	0.172	0.095
News popularity	Lin. Reg.	0.1	0.194	0.185	0.175	0.096

Table 2: Comparison of the Approaches: Error Evaluation.

Figure 4 illustrates the average misclassification error of the algorithms when producing logistic regression (left), SVM (center) classifiers, and the mean squared error when producing linear regressors (right). The figure reports the errors when varying the number of sharers in the aggregation scheme. It uses the Census Income dataset for the classification tasks and the News popularity dataset for the regression task. The shaded areas denote the 95% confidence intervals. The dotted dark orange lines (NP-Pred) illustrate the prediction results obtained using a *non-private predictor* on the whole training set, thus represent the best result attainable.

The results show the following trends. (1) All methods considered produce, in general, predictors that outperform the single agent predictors (NP-Agt). The only exception is given by the simple scheme adopted by PFDS(s) for the logistic regression tasks and when less than half of the agents share their data. This behavior is explained by the lack of coordination of this multi-agent scheme, in which agents may produce privacy-preserving data whose target labels may be representative of the agent's data exclusively, rather than the whole dataset. (2) Both DP-Pred and PFDS produce classifiers that reduce the errors as the number of sharers increases, and plateau when about half of the agents share their privacy-preserving data or privacy-preserving classifier. On the other hand, PFDS(s) appear having more irregular trends, especially on classification tasks. This behavior again is explained by the lack of multi-agent coordination of this approach. These results thus support the hypothesis that using shared privacy-preserving predictors to assign one data labels may be advantageous to reduce

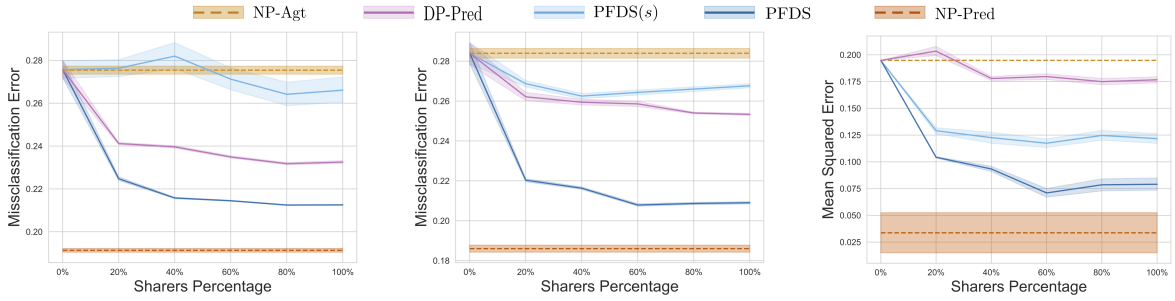

Figure 4: Logistic regression (left), SVM (center), and linear regression (right) predictors for Census and News popularity data.

the overall prediction error. (3) Furthermore, in all experiments, PFDS outperforms its competitors bringing an error reduction of up to 4.4% for classification tasks and 9.7% for regression tasks when compared to DP-Pred. Both versions of PFDS exceed DP-Pred predictions on the regression tasks. This is because the noise introduced by the functional mechanism depends on the dataset dimensionality (which is 61 for the Online News Popularity dataset), while PFDS noise depends only on the maximum tree depth used during partitioning and data generation steps.

Table 2 tabulates the errors produced by the algorithms for all the datasets and for different privacy budgets $\epsilon = 1.0, 0.5$, and 0.1, when $\mathcal{S} = [m]$. The results further strengthen the above observations. In particular, PFDS dominates its competitors in all settings, bringing average error reductions of up to 5.3% for logistic regression, 8.1% for SVM, and 11.3% for linear regression tasks, when compared to aggregated privacy-preserving predictors (DP-Pred). *Additionally, PFDS provides the advantage of sharing datasets, rather than predictors. Datasets are not tailored to a unique prediction task and therefore can be adopted for additional analytic tasks.*

5 RELATED WORK

Several methods have been proposed to minimize the empirical loss function $J(\rho_{\mathbf{w}}, D)$ while satisfying differential privacy. The *functional mechanism*, proposed by Zhang et al. [29], perturbs the objective function by first approximating it using a polynomial, and then perturbing every coefficient of the polynomial with Laplace noise. When applied to linear regression, the scale of Laplace noise is $2(1 + 2(d + 1) + (d + 1)^2)$, where d is the number of dimensions of the dataset. This method has also been extended to handle other regression tasks, where the objective function is not a polynomial with finite order. The extension [29] relies on using the first two terms of the Taylor expansion to approximate the objective function. In this paper, the experiments compared PFDS against a differential private linear regression model obtained through the functional mechanism. In addition to the functional mechanism, the *objective perturbation* [4] was proposed to achieve privacy by adding noise directly to the objective function of the ERM task. To do so, Chaudhuri and Monteleoni [4] showed that several ERM tasks can be performed with a differential private algorithm that minimizes a perturbed objective: $\tilde{J}(\rho_{\mathbf{w}}, D) = J(\rho_{\mathbf{w}}, D) + \frac{\beta^T \mathbf{w}}{|D|}$, where β is a random vector sampled from the Gamma distribution with shape parameter n and rate $\frac{2}{|D|\lambda\epsilon}$. The experiments compared PFDS against differential private logistic regression and SVM models obtained via the objective perturbation.

A third approach, most closely related to PFDS, is to publish a privacy-preserving version of the original dataset, often from a synopsis of the dataset in the form of a nosy histogram. The output of a predictor applied to a differential private dataset will also be differential private by post-processing immunity of differential privacy (Theorem 2.3). Publishing a synopsis enables additional exploratory and predictive data analysis tasks to be performed, and therefore it is often preferred to publishing a privacy-preserving model which is specific to a single task. Mohammed et al. [21] uses a procedure, called *DiffGen*, to generalize and partition the data attributes according to a given taxonomy and applies random noise to the true count of each group of entries.

A limitation of DiffGen is that it relies heavily on the existence of a taxonomy tree, which may not be available in general. PFDS was compared against DiffGen for classification tasks and to estimate the data fidelity of the generated dataset. Other approaches for data partitioning exists. However, they mainly focus on low dimensional data. For instance, the hierarchical mechanism of Hay et al. [16] and its extensions [5, 13, 25] produces an histogram by enforcing additive constraints based on a tree structure of the data universe. In contrast to these related work, PFDS targets a multi-agent setting for prediction rather than the release of a single dataset.

6 CONCLUSION

This paper proposed a Privacy-preserving Federated Data Sharing (PFDS) protocol that each agent can run locally to produce a privacy-preserving version of its original dataset. PFDS is composed of two phases. In a first phase, an agent builds a locally trained privacy-preserving predictor that is shared with other agents. In a second phase, the agent builds a privacy-preserving version of its data using the collected predictors to generate its labels, leveraging the knowledge transferred from all shared models. The PFDS protocol was evaluated on several standard prediction tasks and the experimental results show that the protocol allows agents to construct predictors which are up to 11% more accurate than existing privacy-preserving ensembles of predictors methods. These predictors leverage the presence of multiple datasets while guaranteeing the privacy of the participating individuals.

REFERENCES

[1] Arthur Asuncion and David Newman. 2007. UCI machine learning repository. (2007).

[2] Eric Bauer and Ron Kohavi. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning* 36, 1-2 (1999), 105–139.

[3] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[4] Kamalika Chaudhuri and Claire Monteleoni. 2009. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*. 289–296.

[5] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. 2012. Differentially private spatial decompositions. In *Data engineering (ICDE), 2012 IEEE 28th international conference on*. IEEE, 20–31.

[6] Bolin Ding, Marianne Winslett, Jiawei Han, and Zhenhui Li. 2011. Differentially private data cubes: optimizing noise sources and consistency. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 217–228.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *TCC*, Vol. 3876. Springer, 265–284.

[8] Cynthia Dwork and Aaron Roth. 2013. The algorithmic foundations of differential privacy. *Theoretical Computer Science* 9, 3-4 (2013), 211–407.

[9] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. 2017. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application* 4 (2017), 61–84.

[10] Hadi Fanaee-T and Joao Gama. 2014. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence* 2, 2-3 (2014), 113–127.

[11] Kelwin Fernandes, Pedro Vinagre, and Paulo Cortez. 2015. A proactive intelligent decision support system for predicting the popularity of online news. In *Portuguese Conference on Artificial Intelligence*. Springer, 535–546.

[12] Ferdinando Fioretto and Pascal Van Hentenryck. 2018. Constrained-based Differential Privacy: Releasing Optimal Power Flow Benchmarks Privately. In *Proceedings of the International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*. 215–231.

[13] Ferdinando Fioretto, Chansoo Lee, and Pascal Van Hentenryck. 2018. Constrained-based Differential Privacy for Private Mobility. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 1405–1413.

[14] Arik Friedman and Assaf Schuster. 2010. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 493–502.

[15] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. 2000. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics* 28, 2 (2000), 337–407.

[16] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 1021–1032.

[17] Simon Haykin. 1994. *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

[18] Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N Wright. 2009. A practical differentially private random decision tree classifier. In *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*. IEEE, 114–121.

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

[20] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*. IEEE, 94–103.

[21] Noman Mohammed, Rui Chen, Benjamin Fung, and Philip S Yu. 2011. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 493–501.

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

[23] MP Perrone. 1993. When networks disagree: Ensemble methods for hybrid neural networks. *Neural Networks for Speech and Image Processing* (1993), 126–142.

[24] Michael Peter Perrone. 1993. *Improving regression estimation: Averaging methods for variance reduction with extensions to general convex measure optimization*. Ph.D. Dissertation. Citeseer.

[25] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Differentially private grids for geospatial data. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 757–768.

[26] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 3–18.

[27] Staal A Vinterbo. 2012. Differentially private projected histograms: Construction and use for prediction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 19–34.

[28] Yonghui Xiao, Li Xiong, Liyue Fan, Slawomir Goryczka, et al. 2014. DPCube: Differentially Private Histogram Release through Multidimensional Partitioning. (2014).

[29] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. 2012. Functional mechanism: regression analysis under differential privacy. *Proceedings of the VLDB Endowment* 5, 11 (2012), 1364–1375.

[30] Xiaojian Zhang, Rui Chen, Jianliang Xu, Xiaofeng Meng, and Yingtao Xie. 2014. Towards accurate histogram publication under differential privacy. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 587–595.