PrivaTree: Collaborative Privacy-Preserving Training of Decision Trees on Biomedical Data

Final version available on IEEEXplore: https://dx.doi.org/10.1109/TCBB.2023.3286274

Yamane El Zein, Mathieu Lemay, and Kévin Huguenin, *Member, IEEE*

Abstract—Biomedical data generation and collection have become faster and more ubiquitous. Consequently, datasets are increasingly spread across hospitals, research institutions, or other entities. Exploiting such distributed datasets simultaneously can be beneficial; in particular, classification using machine learning models such as decision trees is becoming increasingly common and important. However, given that biomedical data is highly sensitive, sharing data records across entities or centralizing them in one location are often prohibited due to privacy concerns or regulations. We design PrivaTree, an efficient and privacy-preserving protocol for collaborative training of decision tree models on distributed, horizontally partitioned, biomedical datasets. Although decision tree models may not always be as accurate as neural networks, they have better interpretability and are helpful in decision-making processes, which are crucial for biomedical applications. PrivaTree follows a federated learning approach, where raw data is not shared, and where every data provider computes updates to a global decision tree model being trained, on their private dataset. This is followed by privacy-preserving aggregation of these updates using additive secret-sharing, in order to collaboratively update the model. We implement PrivaTree, and evaluate its computational and communication efficiency on three different biomedical datasets, as well as the accuracy of the resulting models. Compared to the model centrally trained on all data records, the obtained collaborative model presents a modest loss of accuracy, while consistently outperforming the accuracy of the local models, trained separately by each data provider. Moreover, PrivaTree is more efficient than existing solutions, which makes it usable for training decision trees with numerous nodes, on large complex datasets, with both continuous and categorical attributes, as often found in the biomedical field.

Index Terms—Decision Trees, Privacy-Preserving Machine Learning, Scalability, Biomedical Data, Decision-making

1 Introduction

Like many others, the biomedical field has witnessed an important and rapid increase in the volume of generated data, such as patient data from hospitals, experimental data from labs, or even data generated by smartphone applications or wearable devices in the context of the Internet of Medical Things (IoMT). A concrete example of the latter would be iOS applications developed using Apple's ResearchKit[1], a framework designed to enable researchers to easily gather medical data at scale. It is evident that this data is key to the advancement of the field and to better treatment of patients. Such data is also one of the pillars on which the model of personalized medicine relies. Through its analysis, researchers and practitioners may uncover new findings, or train predictive machine learning models to assist in critical decision-making processes such as disease diagnosis.

Building robust predictive machine learning models requires access to large datasets [1]. In many situations, data is spread across several entities. As examples, one can think of a medical datasets held by multiple hospitals, or IoMT devices generating the same types of data, but belonging to different owners. Whenever the common goal of these entities is training predictive machine learning models on their data, all involved parties could benefit from combining their datasets and exploiting all available data at once. Doing so would increase the amount of training data and its diversity, and as a result, enhance the trained model's generalizability.

One straightforward solution to obtaining a model that benefits from all available data would be to centralize all data records in one location, and exploit them as one unified dataset. However, data centralization gives rise to serious privacy issues when the data in question is of sensitive nature, as is the case with biomedical data. This is due to the fact that the storage and use of this data is no longer controlled by its owner. Another solution would be to share the data points among the different data providers. However, often, data sharing is restricted due to strict privacy concerns and regulations. Additionally, different data providers could be in different jurisdictions, which further complicates the data sharing process. To remedy the limitations that arise due to privacy concerns and restrictions, federated learning (FL) [2, 3, 4, 5], a decentralized machine learning training approach through the exchange of model parameters, has emerged as one solution that allows the collaborative training of machine learning models between different data providers without them having to disclose their datasets to third-parties or to each other. Solutions based on federated learning currently mostly target the training of neural networks or ensemble decision tree mod-

- *Y. El Zein is with CSEM SA, Neuchâtel, Switzerland, and the Department of Information Systems at the University of Lausanne, Switzerland. E-mail: yamane.el-zein@csem.ch*
- *M. Lemay is with the CSEM SA, Neuchâtel, Switzerland.*
- *K. Huguenin is with the Department of Information Systems at the University of Lausanne, Switzerland.*

1. https://www.apple.com/lae/researchkit/

els such as gradient-boosted decision trees, as confirmed by the works surveyed in [5]. There has been some effort in the biomedical community to design FL-based solutions to problems such as drug discovery [6], or transciptome-based single-cell annotation [7]. Other solutions rely on the use of cryptographic primitives, such as secure multiparty computation or homomorphic encryption. For example, Ma et al. [8] use secure multiparty computation for privacy-preserving drug discovery. However, due to their important computation and communication overheads, such solutions can be difficult to scale to large datasets [9].

Given that we are interested in the analysis of biomedical datasets for decision support, we focus on a particular type of machine learning algorithm: decision trees. Decision tree models, although not always as accurate as other types of machine learning models such as neural networks, have many advantages. The most notable one is model interpretability and explainability, which is essential in the medical domains, where decisions must be accompanied by clear explanations [10, 11]. Such explanations are less straightforward with black-box models such as neural networks. As such, decision trees have been extensively used in the biomedical community [12, 13, 14, 15]. Another advantage is the fact that decision trees are relatively fast to train. Further, such models offer a certain flexibility, as multiple decision trees may also be combined into ensemble models, such as random forests [16] or gradient-boosted decision trees [17], in order to improve model generalizability and predictive power by reducing over-fitting, at the expense of interpretability.

We tackle the problem of decision tree training in the setting where multiple data providers such as hospitals, research institutions, or IoMT device owners, holding private datasets, aim to train a single decision tree model from the combination of their datasets, without disclosing their records to each other. We focus on the case where the dataset is horizontally partitioned across data providers, meaning that different data providers hold different sets of data records with the same attributes.

1.1 Contributions

We propose PrivaTree, a practical and scalable privacy-preserving protocol, tailored for biomedical applications, with minimal communication and computation overheads, for collaboratively training decision tree models on horizontally partitioned data with both continuous and categorical attributes. PrivaTree allows for collaborative learning without centralization of the raw data of all involved parties in one location, and does not require heavy cryptographic operations on the data records to guarantee privacy. Instead, our approach is similar to that proposed by Shokri and Shmatikov [2] for neural networks, whereby each data provider acts as a local learner during the model training process. During execution of the protocol, each local learner computes updates for growing the decision tree model that is being trained, on their private data records, and securely aggregates these updates with those of other data providers. Computing these updates is equivalent to finding the best way to split a decision tree node to further grow the decision tree model.

We evaluate our protocol for different numbers of data providers, on three publicly available biomedical datasets. Our results show that the decision tree models obtained using PrivaTree present a modest loss in accuracy compared to models trained directly on the entire set of data records, centralized at one location, without any privacy considerations. Moreover, the accuracy of our model consistently outperforms the non-collaborative local models, independently trained by each data provider, exclusively on their private dataset, without any collaboration. Compared to prior work, PrivaTree is substantially more efficient in terms of computation and communication costs, which can be essential for time-sensitive applications. When training a decision tree on data from 4 data providers, on the first dataset with 1688 records and 16 attributes, PrivaTree runs in 145 seconds and its total communication cost is 317 kB. On the second dataset with 11 984 records and 14 attributes, it runs in 130 seconds and the total communication cost is 287 kB. Finally, on the largest dataset with 56 553 records and 21 attributes, PrivaTree runs in 1.56 hours, and requires 12.1 MB in communication.

The results show that PrivaTree is suitable for training decision trees with numerous nodes, on large complex datasets, with both continuous and categorical attributes.

2 Related Work

Much of the existing work in privacy-preserving machine learning relates to the inference (prediction) step, rather than to the training step, as the cryptographic primitives typically used to achieve privacy do not scale well with the large number of operations required for training models. Some of the existing work in privacy-preserving inference specifically targets decision tree models [18, 19, 20, 21, 22, 23]. Bai et al. [22] rely on a secret sharing-based scheme to achieve privacy-preserving decision tree inference in sublinear communication complexity. Similarly, Zheng et al. [23] utilizes additive secret sharing to achieve the same task, outperforming prior work by 4 orders of magnitude in terms of computation, and 163 times in terms of communication.

In this work, we are interested in privacy-preserving training, rather than inference. Shokri and Shmatikov [2] and McMahan et al. [3] propose practical solutions for private collaborative deep learning. The approach in [2] consists of each data provider training a local model, and periodically uploading subsets of their parameter updates to a server, which aggregates them. Data providers can then incorporate them into their local models. The approach in [3] is similar, but the aggregation consists of model averaging. Such approaches, surveyed in [5, 24], are termed *Federated Learning (FL)*. They are substantially more efficient than solutions based on cryptographic techniques. The survey by Nguyen et al. [25] confirms that FL is gaining traction in the healthcare domain.

Our work falls under the federated learning umbrella, as the training data is split across multiple sources, who never disclose their data records. We take interest in decision trees rather than deep learning models, as the high interpretability of decision trees is key for many biomedical applications, where data privacy is also of utmost importance. Recent works on privacy-preserving training

of decision trees are surveyed in [26, 27]. Some focus on privacy-preserving training of tree ensembles, such as random forests [28, 29, 30, 31, 32] and gradient-boosted decision trees [33, 34, 35]. Feng et al. [34] rely on homomorphic encryption to ensure security, while Li et al. [35] relax some privacy constraints and adopt an approach solely based on federated learning, in order to gain in efficiency. Many of the aforementioned works [29, 30, 31] do not target a setting where data is held by different parties, but rather aim at centrally training differentially private random forests.

In this work, we are interested in training single decision trees, rather than ensemble models, in the setting where the data is held by multiple data providers. Such models have the advantage of being more interpretable than ensembles.

Some solutions exist for privacy-preserving training of single decision trees. Several of them aim at privately building decision trees from distributed data which is vertically partitioned (by attributes), by relying on either homomorphic encryption [36, 37] or secure-multiparty computation techniques such as arithmetic secret sharing [38]. Our solution is targeted towards the setting where the data is horizontally partitioned (by records). The works we describe next consider a similar configuration.

Truex et al. [39] propose a hybrid approach based on both differential privacy and homomorphic encryption for training decision trees in the multiparty setting. Different data providers compute counts summarizing their private datasets, add noise to these counts to satisfy -differential privacy, encrypt the noisy results and send them to a central coordinator, who computes the best split to grow the tree on the encrypted noisy responses from all data providers. Although this approach provides good privacy guarantees, it only allows for categorical attributes in the datasets. Moreover, the addition of noise greatly affects the collaborative model's F1-score.

Another solution proposed by de Hoogh et al. [40] is based on Shamir secret-sharing [41]. Similar to [39], it only considers categorical attributes in the training dataset. Abspoel et al. [42] describe a protocol for securely training decision trees on both categorical and continuous attributes when the training data is secret-shared across multiple parties. They propose a method for obliviously sorting the data which reduces the number of operations needed when building the decision tree, thus reducing the overheads of computation on secret-shared data. However the overheads remain substantial for computing a single split in the tree, which becomes problematic when building trees of large depths. We propose a more efficient solution so that training deeper trees becomes accessible.

In summary, our contribution allows for scalable distributed training of decision trees, using both categorical and continuous attributes, when data is horizontally partitioned across data providers, while ensuring data privacy. We propose a federated learning technique targeted towards decision tree models, rather than deep learning models. Compared to existing solutions, it achieves an attractive balance between data privacy, model accuracy, and practical applicability relating to the required computation and communication resources.

3 Background

We provide some background on decision trees, the machine learning model of interest in this work, as well as on federated learning and additive secret-sharing, techniques that we utilize as part of our proposed solution.

3.1 Decision Trees

Decision trees are predictive models, widely used for both regression and classification problems. As their name indicates, the models are structured as trees, composed of nodes and edges. The root node and internal nodes represent a predicate test on a dataset attribute. For categorical attributes, the test corresponds to the equality of the attribute to one of the possible values (categories) that the attribute can take, while for continuous attributes, it corresponds to whether or not the attribute value is larger than a threshold t. The leaf nodes represent a class label for the target variable of the dataset. We refer to the former as decision nodes, and to the latter as classification nodes. The edges represent the outcome of the predicate test of decision nodes. Figure 1 shows a generic example of a decision tree.

During training, every node encompasses the subset of training records that satisfy the predicates of all decision nodes on the path that starts at the root node, and leads to the node of interest. Several algorithms exist for decision tree training, including ID3 [43], and its widely used successor, C4.5 [44], by Quinlan. CART [45] is another popular algorithm, similar to C4.5, but geared towards regression problems. As our solution is targeted towards classification problems, we choose the C4.5 decision tree training algorithm as the base for our privacy-preserving decision tree training protocol.

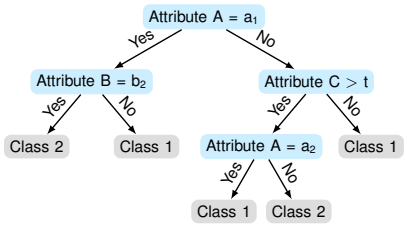

Fig. 1. Example of the structure of a decision tree. A, B and C are dataset attributes. A and B are categorical, and respectively take values in $\{a_1 \ldots a_l\}$ and $\{b_1 \ldots b_m\}$, while C is continuous.

3.1.1 C4.5 Algorithm

The C4.5 training algorithm adopts a greedy top-down approach for building the tree. Beginning at the root node and ending at the leaves, nodes are split with the goal of maximizing "purity" of the resulting child nodes, which refers to the homogeneity of the classes of data records encompassed by that node. C4.5 takes as input a dataset D with M attributes, a_1 to a_M, one target categorical variable Y which take value in $fc_1 \ldots c_k g$ and R data records. These attributes may be either categorical or continuous. A common hyper-parameter, provided as input to the decision tree training algorithm, is a maximum tree depth value, which we denote by max_depth. It ensures that the trained tree does not grow beyond a certain depth and thus helps

mitigate over-fitting of the decision tree model. The algorithm outputs a decision tree. Starting from the root node of the decision tree, which encompasses the entire set of data records, the algorithm selects the attribute that either maximizes the information gain criterion [44], or minimizes the Gini index criterion [45]. The selected attribute is then used for splitting the root node and the associated dataset. This process is recursively repeated on the child nodes created by the splitting process, until one of the stopping criteria described below is reached. In order to select the threshold t to split continuous attributes on, the typical approach is to test the different values taken by the continuous attribute, and select the one that maximizes information gain (resp. minimizes the Gini index). For simplicity, in the remainder of this paper, we choose information gain as the splitting criterion, and provide some notes on how the protocol that we describe in Section 5 would be simply adapted if the Gini index were to be chosen.

The algorithm stops the splitting process if either of the following conditions apply to a node: (1) all data records corresponding to the node belong to the same class, none of the attributes result in a positive information gain criterion value, the depth of the node is equal to the pre-defined *max_depth*.

3.2 Federated Learning

The term "federated learning" was introduced by McMahan et al. [3], in reference to the training of a common machine learning model across multiple data providers, without data leaving the providers' devices or servers. This is achieved through the sharing of training parameters across participants during the training process, rather than sharing data records. One major challenge with the federated learning paradigm is the leakage of information about the underlying data from the exchange of training parameters [46]. In this work, we adopt the federated learning paradigm for its efficiency, while minimizing information leakage through parameters with the use of additive secret sharing.

3.3 Additive Secret Sharing

Additive secret sharing [47, 48] allows multiple parties, each holding a secret input, to perform arithmetic operations on their joint inputs, without revealing them to each other nor centralizing them in one location, which makes it suitable for protecting data providers' parameters in the federated learning setting considered in this work. It does not require the existence of a trusted third party, which is often a strong assumption in practice. As our protocol is geared towards the training task, which typically involves a large number of operations, we choose additive secret sharing over multiparty homomorphic encryption [49] for protecting the data providers' inputs, as the computational and communication overheads of homomorphic encryption schemes remain impractical [9]. Moreover, compared to secure aggregation based on trusted execution environments (TEEs), additive secret sharing does not rely on strong trust assumptions [50], which may not always hold true in practice. Finally, another alternative to additive secret sharing would be Shamir's secret sharing [41], which enables the reconstruction of intermediate results by only a subset of data providers rather than requiring all of their inputs. Even though this provides the protocol with fault-tolerance, the added computational cost is not justified in the setting that we consider for this work, as we deal with a small number of data providers rather than millions of IoT devices, and assume that all participants remain online during the protocol. Further details are provided in Section 5.3.2.4.

We provide details on performing addition using additive secret sharing, as it is the only operation that we carry out using this technique.

Every party splits their secret input $s \in \mathbb{Z}_q$, into N random shares, s_1 to s_N, where N is the number of participants in the protocol. $N-1$ of these shares are chosen uniformly at random in $\mathbb{Z}_q$, while the last share is computed as:

$$s_N = s - \sum_{i=1}^{N-1} s_i \mod q \tag{1}$$

This guarantees that:

$$\sum_{i=1}^{N} s_i = s \mod q \tag{2}$$

As each share is selected uniformly at random in $\mathbb{Z}_q$, access to less than N shares does not reveal information about the secret. Moreover, in order to reconstruct the secret, an adversary must get access to all N shares, and thus must control all involved parties.

Each party then sends one of their shares to every other party, and retains the remaining share.

Let's consider two secret-shared inputs s and t. Assuming N parties are involved in the computation, s is split into shares s_1 to s_N, and t is split into t_1 to t_N. The i^{th} share of each input is distributed to party i. In order for the parties to jointly compute the sum of s and t, every party i locally computes the sum u_i of their shares as follows:

$$u_i = s_i + t_i \mod q \tag{3}$$

Every party shares their locally computed sum with the others. The sum u of s and t can then be computed by every party as:

$$u = \sum_{i=1}^{N} u_i \mod q \tag{4}$$

4 System and Threat Model

In this section, we first define the different entities of our system model, as well as their roles and the relationships between them. We then describe the threat model considered in this work.

4.1 System Model

We consider $N \geq 2$ data providers, who aim to collaboratively train a decision tree model that benefits from their joint data records, without disclosing the latter to each other. We do not consider large-scale federated learning with thousands of edge devices, but rather a few institutions (e.g. hospitals, research centers, etc.) who wish to collaborate while maintaining data privacy.

We consider horizontally partitioned data across data providers, which means that the latter hold different data

records for the same set of attributes and target variable. Our training protocol also calls for the assistance of a central coordinator, which we refer to as "coordinator" in the remainder of this paper. The coordinator does not contribute any data to the training. Its role is to orchestrate the training process by instructing the data providers which tree node is to be split at every step.

We note that the role of the coordinator could be taken by either one of the data providers. For the sake of clarity, we portray it as a separate entity. Figure 2 depicts the considered system, as well as the different communication channels between them.

We assume that the attributes list in the data providers' datasets, the possible values (categories) that may be taken by categorical attributes, an upper and lower bound on the values that continuous attributes can take, and the different classes of the target variable are known to data providers and to the coordinator.

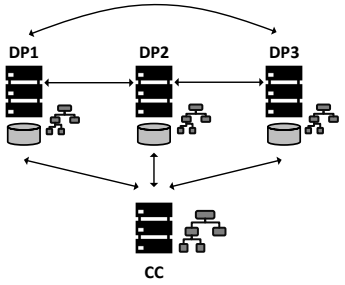

Fig. 2. Overview of the system's entities and communication channels. DP stands for "Data Provider" and CC for "Central Coordinator." The arrows represent secure communication channels. We portray the CC as a separate entity for the sake of clarity.

4.2 Threat Model

We consider all entities to be honest-but-curious, meaning that they will not deviate from the agreed-upon protocol, but could still attempt to infer information about other entities' private data from messages received during the correct execution of the protocol. We discuss the implications of having malicious and colluding actors in Section 8.5. We do not focus on privacy leakage from the resulting model through attacks such as model inversion [51] or membership inference [52]. These may be addressed within the framework of differential privacy [53], at the expense of an important drop in model accuracy. Further, frameworks such as the one proposed by Zhu and Du [54] may be used to quantify the leakage resulting from the fact that the resulting trained model is available to all data providers at the end of the protocol. Finally, we assume that secure point to point communication channels exist between all system entities.

5 Protocol

The goal of PrivaTree is to enable multiple data providers to train a decision tree, up to its optimal depth, while keeping the underlying training records private. We provide an overview of PrivaTree, and detail its two phases.

5.1 Protocol: Overview

PrivaTree falls under the umbrella of federated learning, whereby multiple parties hold subsets of the training data, and train a common model without exchanging data points. Computations that directly touch a data provider's private dataset are carried out only by that data provider. The results are securely aggregated across data providers, to reduce information leakage through intermediate results, using additive secret-sharing. The aggregated results are used to update the collaborative model. The two phases of PrivaTree are described in sections 5.2 and 5.3. Algorithms 2 and 3 summarize the steps taken by the coordinator and the data providers, respectively.

5.2 Protocol: Max-Depth Determination Phase

The goal of the max-depth determination phase is to collaboratively compute a maximum depth value for the decision tree. When training a decision tree on a centralized dataset, this value is typically determined by using randomized grid-search k-fold cross-validation on the training dataset [55]. Since our training set is split across different data providers who cannot access each other's data records, we approximate the optimal value of this hyper-parameter. We include this hyper-parameter selection step as an integral part of the PrivaTree protocol to avoid the tree depth to be chosen arbitrarily prior to training. In fact, choosing a tree depth that is too small for the data at hand would lead to a model that underfits the data and thus results in less accurate predictions on both the training and test sets, while choosing a tree depth that is too large would overfit the training data, and lead to inaccurate predictions on unseen data.

Our approach is as follows: each data provider computes the optimal tree depth on their local dataset, using randomized grid search k-fold cross-validation. This step can be performed separately by every data provider, prior to the online phase of the protocol, as it does not requires any communication between them. The data providers then securely compute the average maximum tree depth value, using additive secret sharing. Every data provider's input to the additive secret sharing protocol is the locally computed optimal maximum tree depth. Each data provider splits this input into secret shares, and distributes them to other data providers. Then, each computes the local sum of their received shares, and sends this locally computed sum to the other data providers. Finally, the sum of optimal maximum tree depth values can be computed by each data provider, by summing the received local sums. This sum can then locally be divided by the total number of data providers to obtain the average maximum depth value, which is shared by one of the data providers with the coordinator, who uses it as parameter for the training phase.

5.3 Protocol: Training Phase

During the training phase, the different data providers, assisted by the coordinator, collaboratively train a decision tree. We first describe a lightweight sub-protocol that use during this phase, before detailing the training protocol.

5.3.1 Secret Selection Sub-Protocol (SSSP)

PrivaTree relies on a sub-protocol that enables data providers to securely find the most voted for element in a list. In this case, this list is composed of either the different attributes in the datasets, or the different class values that the target variable can take. We note that this sub-procotol has the same goal as the protocol proposed by Bonawitz et al. [56]. However, the SSSP that we propose allows us to forego the need for a central coordinator, as well as work in groups of small size, thus reducing overheads.

We assume N parties would each like to select one element out of a list of M elements, without revealing their selections, and find the element selected by a majority of parties. Every party p_i locally creates a mask v_i, a vector of length M representing the different elements to choose from, where all elements $v_{i;j}$, are set to 0 except the one selected by p_i, which is set to 1. For every value of $j \in \{1; \ldots; M\}$, the parties compute, using additive secret sharing, the sum $\sum_{i=1}^{N} v_{i;j}$. The element with the largest sum is the most selected one. We note that the modulus for the additive secret sharing step of this sub-protocol can be as small as $N + 1$, since the only possible values of the input to the addition operation are 0 or 1. Therefore the maximum value of their sum is N. We favor smaller modulus values to reduce the size of exchanged messages. The SSSP is summarized in Algorithm 1.

Algorithm 1 Secret Selection Sub-protocol (SSSP)

Input: List L of M elements to select from.
Output: Element m of L selected by majority of voters.

1: **for all** voters p_i **do**
2: Select preferred element m_i from L
3: Create a mask v_i of length $|M|$
4: Set the entry in v_i corresponding to m_i to 1, and all other entries to 0
5: **for all** $v_{i;j}$ in v_i **do**
6: Create secret shares of $v_{i;j}$ and distribute them across voters
7: $S \leftarrow Vector(size : |v_i|)$
8: **for all** mask entries j **do**
9: Receive vector of secret shares s_j of j from other voters
10: $S[j] \leftarrow sum(s_j)$
11: $k \leftarrow \operatorname{argmax}_{s \in S} S$
12: $m \leftarrow M[k]$
13: Output m

5.3.2 Training Protocol

Every data provider starts with a local decision tree model, and the coordinator starts with a global model, each with only a root node.

5.3.2.1 Node Splitting: At every iteration of the protocol, the coordinator selects an unprocessed tree node at random, and instructs the data providers that it should be split. Each data provider then computes, on their private dataset, for the chosen node, the information gain values (resp. Gini index) of all attributes, and selects the attribute with maximal information gain (resp. minimal Gini index). If a certain data provider cannot split the node (See stopping criteria 1-2 in Section 3.1.1), it selects the special value `no_split` instead. The data providers then engage in the SSSP to find the attribute selected by the majority. The list to make a selection from contains all attributes of the dataset, as well as the `no_split` special value. In case of ties, a second round of voting takes place, including only the most voted for attributes of the first round. If a tie occurs again, an attribute is chosen at random. The most voted for attribute is then shared by one of the data providers with the coordinator. If most data providers choose the `no_split` special value, the coordinator instructs the data providers to skip splitting the node in question and to collaboratively select a class label for it. Otherwise, if the most selected attribute is categorical, the coordinator provides information about how to split the node into child nodes depending on the category values that the selected attribute can take. This is to ensure that all data providers and the coordinator have the same tree structure and node indexing for the rest of the training process. The data providers then update their local decision tree models accordingly, by splitting the data records of the selected node, based on the collaboratively obtained predicate. The coordinator also splits the selected node in the global model. Finally, if the chosen attribute is continuous, the coordinator instructs the data providers to select an appropriate threshold for splitting the tree node. The coordinator splits this node into two child nodes, one corresponding to values above a threshold, and another to values below it. We note that the value of this threshold is irrelevant and unknown to the coordinator. It is only known by the data providers.

5.3.2.2 Threshold Selection for Continuous Attributes: When the attribute selected is continuous, a threshold must be specified for splitting the decision tree node. Given the node to split and the collaboratively chosen attribute, each data provider finds the best threshold for splitting the node using that attribute, on their local dataset. Then, the data providers use additive secret-sharing, to compute the sum of their thresholds, and in turn their average value by locally dividing the obliviously computed sum by N. The use of additive-secret sharing ensures that the threshold value of every data provider remains private, as it could leak information about the data distribution. As the thresholds computed by the data providers may not always be discrete, we quantize them before using them as secret input to additive secret sharing. If a data provider cannot compute a value for the threshold due to the fact that, on their private dataset, no data records correspond to the node being split, the data provider's input to additive secret sharing is the mid-range of the attribute chosen by most data providers to split that node. In most cases, this value would minimally affect the result of the computation of the average threshold. This obfuscates the fact that the data provider cannot compute the threshold on their private dataset. Figure 3 describes the messages exchanged between data providers and the coordinator for splitting one decision tree node.

5.3.2.3 Leaf Node Labeling: When most data providers choose not to split a node, it must be assigned a class label. Every data provider computes the best class label locally, for their dataset. The data providers engage in the secret-selection sub-protocol to determine the class

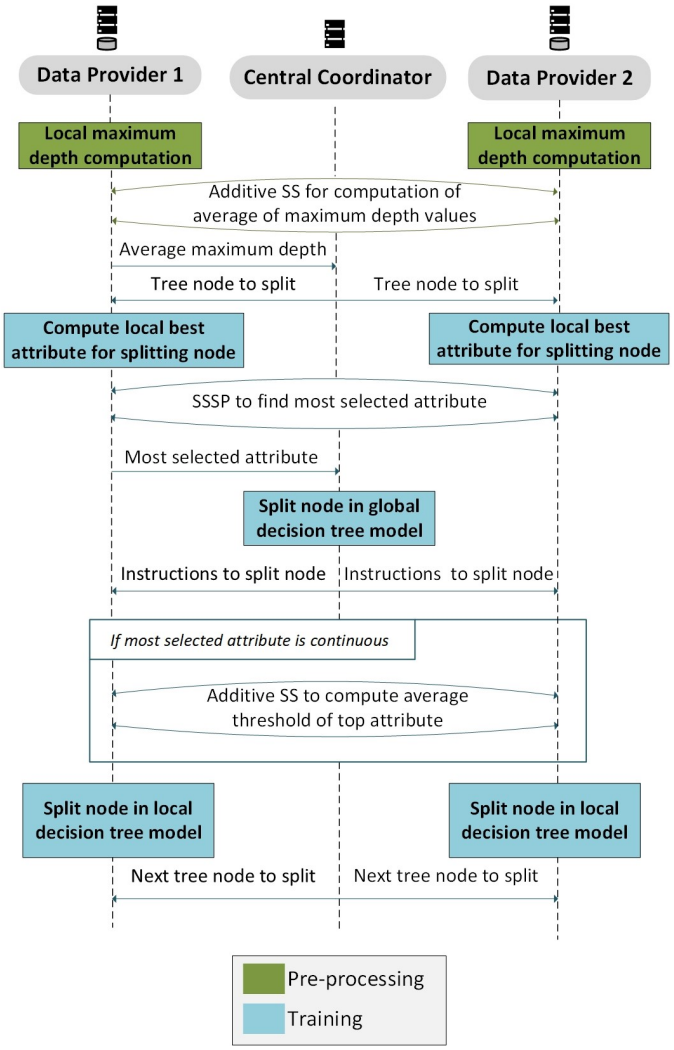

Fig. 3. High-level representation of PrivaTree when splitting one node. We represent only 2 data providers for clarity.

chosen by the majority. Each data provider assigns this class label to the node in their local model.

5.3.2.4 Participants Going Offline and Message Loss: As our protocol is designed to be used by a small number of willing participants rather than millions of mobile devices, we assume that disconnection of a participant and message losses are rare events. If any of the latter occurs, online participants can detect it when they do not receive all expected messages at the on-going step of the protocol, within a pre-defined time interval. We consider that during training, all data providers and the central coordinator keep a copy of the current tree structure, as well as the index of the current node of the tree that is being processed, which persist after disconnection. When one or more participants drop out, the first participant that detects the anomaly instructs all others to exit. The participants then reconnect to each other, and resume the training starting from the saved node index and tree.

5.3.2.5 Termination: PrivaTree terminates when all nodes have been processed. We consider a node to be processed when either of the following applies: (1) the node was split using a collaboratively chosen attribute, (2) the node was labeled with a class after a majority of data providers chose not to split it, (3) the depth of the node has reached the *max_depth* value.

In such a case, the coordinator instructs the data providers to terminate. Each data provider's local tree model is complete (and equivalent to that of other data providers). The coordinator's global model is a minimal version of the data providers' models. The tree has the same structure, but lacks the threshold values of decision nodes relative to continuous attributes, as well as the class labels of classification nodes. These values are directly assigned by each data provider to their model, and not shared with the coordinator. This minimizes the number of exchanged messages, and does not change the outcome of the training as this information is not needed by the coordinator, since the global model held by the coordinator is only used for coordination, and not for classification. Following termination, each data provider can use their trained local tree model to classify new data points.

Algorithm 2 Coordinator

Input: `max_depth` computed in max-depth determination phase

1: Send randomly selected `node` to data providers
2: Receive `top_attribute` selected by majority of data providers
3: **if** majority of data providers vote `no_split` **then**
4: Send `skip` to all data providers and label it *f* tells data providers not to split `node` *g*
5: Go to step 12
6: **if** `top_attribute` is continuous **then**
7: Add 2 children to `node` for above and below threshold
8: Instruct data providers to compute threshold
9: **else if** `top_attribute` is categorical **then**
10: Split `node` of global tree using `top_attribute`
11: Instruct data providers how to split the node *f* order of children *g*
12: Mark `node` as processed
13: **if** no more unprocessed nodes exist **then**
14: Send `stop` to all data providers and exit
15: Randomly select unprocessed `node`
16: Repeat steps 1 to 16 on new node

6 Evaluation

In this section, we provide the details of our implementation and experimental setup, and describe the datasets used for evaluation.

6.1 Implementation

We implement PrivaTree in Python, and evaluate its computation and communication costs, as well as the accuracy of the collaboratively trained decision trees. We model the coordinator and data providers as peer-to-peer (P2P) nodes in a P2P network. We rely on the `p2pnetwork` [57] package (v1.1) for the implementation of the P2P functionalities. We take advantage of the computational efficiency of the `scikit-learn` [58] machine learning library (v1.0.1),

Algorithm 3 Data provider

Input: Local dataset
1: Set `local_node` = root of local tree
2: Receive `node` from coordinator
3: Set `local_node = node`
4: **if** stopping criteria (1) or (2) is locally satisfied **then**
5: `attr_selected = no_split`
6: **else**
7: `attr_selected = max_gain(local_node)`
8: `global_best_attr` SSSP with other data providers
9: Send `global_best_attr` to coordinator
10: **if** `global_best_attr is continuous` **then**
11: Locally compute threshold `local_th`
12: `avg_threshold` additive SS with other data providers
13: Split local `node` on `global_best_attr` and `avg_threshold`
14: **else if** `global_best_attr` is categorical **then**
15: Receive splitting instructions from coordinator $f$order of children$g$
16: Split `local_node` according to instructions
17: **else if** `global_best_attr` is `no_split` **then**
18: `node_label` SSSP with other data providers
19: Label `local_node` with `node_label`
20: **if** `stop` is received from coordinator **then**
21: Return local tree and exit
22: Repeat steps 2 to 22

specifically for the selection of the optimal maximum tree depth by the data providers (Section 5.2), which can be computationally expensive with large datasets. We also make use of the `scikit-learn` library for the top attribute selection step taken locally by each data provider.

6.2 Experimental Setup

For our experiments, we fix the following parameters. During the max-depth determination phase (Section 5.2) of the protocol, the data providers use randomized grid-search k-fold cross-validation to determine the optimal maximum depth on their private datasets. We set $k = 5$. During the training phase, when collaboratively computing the best threshold to split a continuous attribute with, using additive secret sharing, the data providers must quantize their inputs. We quantize the values to 16-bit integers. Based on the assumption that upper and lower bounds on continuous attribute values are known to all data providers (Section 4.1), we execute the additive secret sharing protocols in $\mathbb{Z}_q$ with q being the first integer larger than $w = N \quad u$, where N is the number of data providers, and u is the largest upper bound on a continuous attribute in the dataset, since the sum of threshold values cannot be larger than w.

We evaluate the accuracy of the collaboratively trained model for $N \ 2 \ f5, 10, 15, 20, 25g$. We split the training instances into random partitions, where each represents the private dataset of one data provider. To account for biases in the generated splits, we create 20 different random splits, and run our experiments for each of them. We report the mean and standard deviation (SD) of the accuracy of the models over the 20 runs. We perform three different types of experiments, which differ in the size r_{dp} of each data providers' partition. For the first type of experiments, all data providers have an equal number of records $r_{dp} = \frac{r}{N}$, where r is the total number of records. For the second type of experiments, all data providers have an equal number of records $r_{dp} = \frac{r}{25}$. This ensure a fixed number of records per data provider regardless of the value of N. For the third type of experiments, r_{dp} differs per data provider. For each value of N, $\frac{N}{5}$ data providers hold 10%, 15%, 20%, 25% and 30% of the data. For each experiment, we run PrivaTree, and compare the model accuracy to that of the central model, trained on the data records of all data providers, and to those of the local model, where single data providers train models separately on their local records. We evaluate the communication and computation costs of PrivaTree, by running it for $N = 4$, across 5 machines, connected over a local area network. One of the machines, with an Intel i7-6700 CPU at 3.4 GHz and using 32 GB of RAM, acts as the coordinator, while the others, each with an AMD 2950X CPU at 3.5 GHz and using 128 GB of RAM, act as data providers.

6.3 Datasets

We select three publicly available biomedical datasets, to evaluate our protocol with. The first one is the "Estimation of obesity levels based on eating habits and physical condition" dataset [59], from the UCI Machine Learning Repository [60]. It is aimed at the prediction of individuals' obesity levels based on their physical condition and eating habits. The second one, also from the UCI Machine Learning Repository, is the "EEG Eye State" dataset. The classification task associated with it is the detection of whether an eye is open or closed based on features extracted from EEG measurements. The third dataset is the "Diabetes Health Indicators Dataset" from Kaggle.[2] It is a cleaned and consolidated version of the "Behavioral Risk Factor Sureveillance System" dataset released by the CDC in 2015.[3] The classification task associated with it is the prediction of diabetes based on survey responses to questions concerning health conditions and behaviours. Table 1 summarizes the sizes and specifications of the three datasets. For simplicity, we refer to the aforementioned datasets as the "Obesity", "Eyes", and "Diabetes" datasets respectively. In our experiments, we use 80% of the records from each dataset for training, and 20% for testing. We clean the datasets prior to using them, by removing features with more than 70% missing values, as well as any remaining records containing missing values.

TABLE 1
Datasets used for evaluation

Dataset	Train	Test	Categorical	Continuous	Classes
Obesity	1688	423	8	8	7
Eyes	11 984	2996	0	14	2
Diabetes	56 553	14 139	14	7	2

As some operations take place in $\mathbb{Z}_q$, we transform attribute values into positive ones, prior to running PrivaTree. data providers add an offset to all values of the attribute,

2. https://www.kaggle.com/
3. https://www.cdc.gov/brfss/annual_data/annual_2015.html

equal to the absolute value of the minimum (< 0) possible value that this attribute can take. This has no effect on information gain computation as the order of values is unchanged. We note that this minimum value is not necessarily the minimum of that attribute across data providers. It is a lower bound on the values that the attribute may take, assumed to be known by all data providers.

7 Results

We report the results of our experiments. We rely on model accuracy to assess the predictive performance of the collaborative model. The provided results are obtained with information gain as the splitting criterion. We observe similar trends when using the Gini index. To assess the efficiency of PrivaTree, we report the total bandwidth, as well as the total run time of PrivaTree. The results are discussed in Section 8.

7.1 Model Accuracy

The average accuracy over 20 runs of the collaborative, central, and local models for different values of N is reported in Figure 4, for the experiments described in Section 6.2. Additional results showing the models' F1 score, precision, and recall can be found in Section 2 of the supplemental material. We discuss the model's performance in Section 8.1.

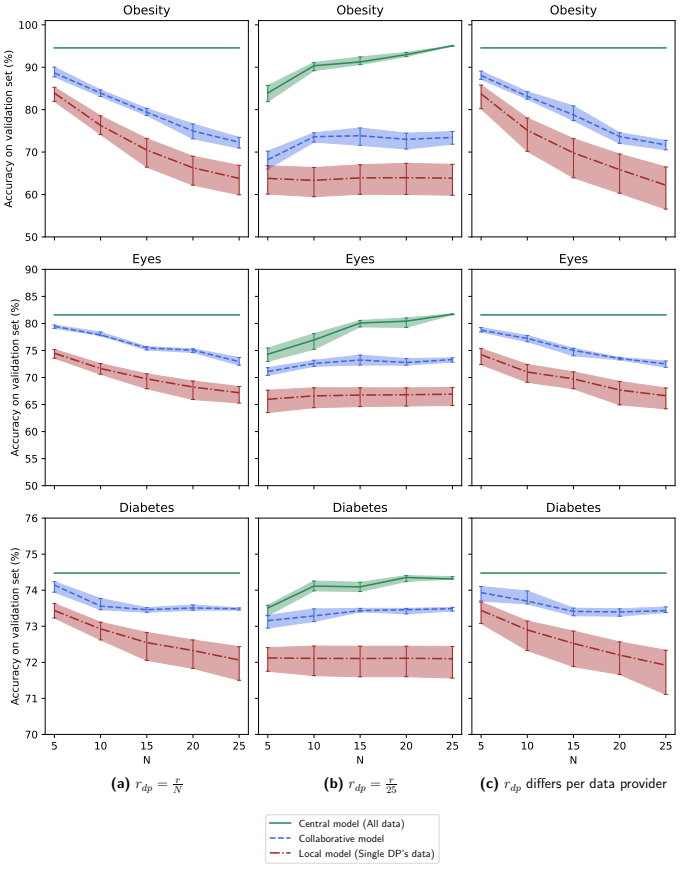

Fig. 4. Accuracy of the central, collaborative, and local models for (a) $r_{dp} = \frac{r}{N}$, (b) $r_{dp} = \frac{r}{25}$, and (c) r_{dp} variable per data provider. Shaded areas and error bars show one SD above and below the mean. Baseline accuracy of the majority class classifier is 18%, 55%, and 49%, for the Obesity, Eyes, Diabetes datasets

7.2 Protocol Efficiency

We execute our protocol in a distributed fashion with 4 data providers and the coordinator. We report the total run time of the online phase of PrivaTree. This includes CPU time of the data providers and coordinator, as well as the communication time between the different participants in the protocol. Additionally, we report the total bandwidth, corresponding to the total size of exchanged messages. The results are summarized in Table 2. As a baseline, we report the CPU time for centralized training (not distributed nor privacy-preserving).

TABLE 2
Protocol run time and total communication costs.

Dataset	Records	Attributes	Tree Nodes	Central CPU Time	PrivaTree Run Time	Bandwidth
Obesity	1688	16	147	0:063 s	145:3 s	317 kB
Eyes	11 984	14	2065	0:609 s	130:4 s	287 kB
Diabetes	56 553	21	32 139	1:078 s	1:56 h	12:1 MB

8 Discussion

We discuss the accuracy, ability to assist in decision-making, efficiency, and privacy guarantees of PrivaTree, and compare our results to related work.

8.1 Model Utility

To ensure both privacy and scalability, our method presents a modest loss in accuracy. When $r_{dp} = \frac{r}{N}$ (Fig. 4 (a)), the accuracy of both the local and collaborative models decreases as N increases, since a larger N means fewer records per data providers. Therefore, accuracy of our collaborative model depends on the number of records per data provider. This is a limitation, as the quality of the collaborative model is bounded by that of the data subsets held by individual data providers. The results in Fig. 4 (b) further confirm this, as the accuracy of the collaborative model grows slower than that of the central model as N increases. The results in Fig. 4 (c) show that when data records are unevenly distributed across data providers, accuracy is similar to when they are equally distributed. The collaborative model systematically outperforms the local one for all three datasets, which means that collaboration using PrivaTree always results in a better model than non-collaboration. Moreover, the SD values for the local model are consistently larger than those for the collaborative model, which suggests more variability in the accuracy of local models depending on the way that data records are distributed. The collaborative model is therefore more stable than the local model. Similar trends are observed for the F1 score, precision, and recall (Section 2 of the supplemental material).

8.2 Assistance in Decision-Making

The ability of tree-based models to help in decision-making is one of the main motivations behind their use in the biomedical field. Hence, it is important for the collaborative model, trained using PrivaTree, to have a structure similar to the privacy-violating central model. We assess the latter

by showing that the collaborative model is able to learn the most discriminative features learnt by the central model. Using the Jaccard index, which ranges from 0 to 1 and represents the similarity between two sets, we compare the unique set of 15 features associated with the top-most tree nodes of the central, collaborative, and local models. The results, summarized in Table 3, show good agreement between the top-most features learnt by the collaborative model and the central model, for all considered datasets and values of N. The results also show that the local model does not perform as well as the collaborative one when it comes to learning the most important features.

TABLE 3
Jaccard index representing the similarity between the top-most features of the central model and the local model (L), and the similarity between the top-most features of the central model and the collaborative model (C), for the Obesity, Eyes, and Diabetes datasets and for different numbers of data providers N. The Jaccard index ranges from 0 to 1. The higher its value, the more similar the top features between two models are. We note that the setting considered is the one where all data providers have an equal number of records $r_{dp} = \frac{r}{N}$.

Dataset	Model	N=5	N=10	N=15	N=20	N=25
Obesity	L	0.449	0.495	0.450	0.481	0.478
	C	0.833	0.833	0.833	0.833	0.666
Eyes	L	0.653	0.531	0.488	0.479	0.391
	C	0.857	0.75	0.75	0.729	0.65
Diabetes	L	0.853	0.727	0.729	0.695	0.665
	C	1	1	1	1	0.8

8.3 Computational and Communication Efficiency

The efficiency of PrivaTree comes from the fact that it makes minimal use of secret-sharing, which is only used for aggregating intermediate training results, rather than for making computations on data records. The bottleneck comes from the rounds of communication required for securely aggregating intermediate results, the number of which is linear in the number of tree nodes, as can be seen in Figure 3. The computation time is thus not directly influenced by the number of records, but rather by the number of nodes in the trained decision tree.

The only operation carried out securely is the addition, which is cheap compared to others on secret-shared data. All other operations are carried out locally, in parallel, by the different data providers. Additionally, since we can compute upper bounds on secret-shared values, we operate in groups of bounded sizes, reducing the size of exchanged messages. For secure aggregation of numeric values (max_depth, splitting thresholds), one secure addition is required. For secure computation of the most selected splitting attribute or the most selected class label, the secret selection sub-protocol is used. It requires F or C secure additions respectively, where F is the number of attributes in the dataset, and C the number of classes. The results in Table 2 confirm that the run time and bandwidth of our protocol depend primarily on the number of tree nodes.

Formally, PrivaTree's complexity can be expressed as $(2^d \quad 1)(O(F\ r_{dp}\ \log(r_{dp})) + O(FN))$, where d is the tree depth, F the number of attributes, r_{dp} the number of records per data provider, and N the number of data providers. The $(2^d \quad 1)$ term represents the number of nodes for a full binary tree. Typically in practice, the number of nodes is lower. The $(O(F\ r_{dp}\ \log(r_{dp}))$ term corresponds to each data provider's attribute and threshold selection, done in parallel with other data providers. It is obtained from the `scikit-learn` documentation, which we make use of to carry out these computations. The complexity of secret-sharing is $O(N)$, since each data provider must compute and send $N \quad 1$ shares [61]. $O(FN)$ represents the secret-sharing step, multiplied by the number of attributes F, since secret-sharing is required F times to select the most voted for attribute (See Algorithm 1). If most data providers choose not to split a node, C secure additions would be needed to label it with the most voted for class label (where C is the number of possible classes). Assuming that $F > C$, we retain the $O(FN)$ in the expression of the algorithm's complexity.

On all three datasets, the run time of PrivaTree is well within practical reach, even when the number of records, features, and nodes in the constructed decision tree are very large, as is the case for the Diabetes dataset.

8.4 Comparison with Prior Work

We compare our solution to that of Abspoel et al. [42], which tackles the same problem of privacy-preserving decision tree training, on horizontally partitioned data, with both continuous and categorical attributes. The authors of [42] propose a method based on secret-sharing data records to 3 or more non-colluding servers, and training the model on the secret-shared records, providing both passive and active security. A key limitation of their work is that trees are grown up to an arbitrary pre-defined depth, limiting the size of the trained decision tree, which leads to underfitting. This is remedied by the authors by combining multiple trees into an ensemble, to approach the accuracy of the central model. However, using ensemble models limits interpretability, which is a key motivation behind choosing tree models for modeling biomedical data. Moreover, the solution is computationally heavy as operations are performed on secret-shared records. Extrapolating from the benchmarks reported in [42] and the function provided for approximating training time, we estimate the run time for training a decision tree with passive security, up to its respective optimal depth $d = 23$, on the Diabetes dataset, using the approach in [42]. We detail this estimation in Section 1 of the supplemental material. We estimate a lower bound of 441 hours (18 days) for training a single decision tree on the Diabetes dataset using the method in [42], while PrivaTree requires only 1.56 hours for the same task. Furthermore, PrivaTree provides a privacy-preserving way to select an appropriate depth, foregoing the need to resort to ensemble models for ensuring good accuracy. This helps maintain interpretability. Moreover, PrivaTree scales well, and is practically usable for training decision tree models with a large number of nodes, on large datasets with both continuous and categorical attributes.

8.5 Privacy

PrivaTree aims at protecting the privacy of the datasets of the different data providers during the collaborative training process. The protocol is private by design, since

at no point does it call for data providers to share their data points with other entities. This is because any computation that must be made on the data records is carried out solely by the data provider who holds the data records in question. PrivaTree does however call for the aggregation of intermediate results of training by the data providers, such as the maximum tree depth, best attribute and threshold to split a specific decision tree node with, or the best class label for a leaf node. These intermediate results can indirectly leak information about the records in a dataset. In order to reduce leakage from these results, we use additive secret-sharing for aggregating numerical results, and the secret selection sub-protocol described in Section 5.3.1 for aggregating qualitative results, also based on additive secret-sharing. Therefore, the privacy guarantees on the exchanged intermediate results is derived from those of additive secret-sharing under the semi-honest threat model, meaning that no data provider can learn anything about other data providers' inputs except what can be inferred from the output of computation on secret-shared values [61]. Further, for every tree node that requires splitting, both the coordinator and data providers only get access to the most voted for attribute, and not to individual votes by data providers. Similarly, the data providers only get access to the average threshold across all data providers, and not to the individually computed best thresholds, while the coordinator receives neither the individual values, nor the aggregated ones. Below, we provide a more detailed analysis of PrivaTree, its privacy guarantees, as well as its limitations.

8.5.1 Exchanged Messages and Intermediate Results

We start by enumerating messages exchanged during execution of PrivaTree, as well as the intermediate results computed by data providers, and describe the privacy guarantees related to them under the non-collusion assumption. In Section 8.5.2, we extend the analysis to the scenario where collusions are possible.

Max-depth Determination and Threshold Selection: The messages sent for these tasks are the shares of locally computed maximum depths (resp. thresholds) by the data providers and the local sums of these shares computed by each data provider. Shares are generated uniformly at random, and thus do not leak information about the maximum depth (resp. thresholds) of each data provider. Local sums are obtained by summing the random shares, thus they do not leak information either. At the end of this step, each data provider learns the sum of depths (resp. thresholds). Given the guarantees of additive secret sharing, the only information that can be inferred is what can be inferred from the sum.

Training Coordination The coordinator sends messages to ensure synchronization between data providers. Such messages are independent from the data providers' records, and thus do not leak sensitive information.

Attribute Selection and Leaf Node Labeling The messages sent for these steps are the secret shares of the attribute (resp. label) selection mask entries and the local sums of the shares. They do not leak information about the mask, as the former are generated uniformly at random, and the latter derived from them. At the end of this step, data providers learn the total number of votes per attribute (resp. label). The individual votes of different data providers are indistinguishable from each other, therefore the information leaked by the aggregated vote cannot be linked to the data records of a specific data provider. Individual data provider votes may only be recovered with certainty if they are all similar. Even in this worst case, the underlying data records remain hidden.

8.5.2 Collusions

Under the no collusion assumption, each data provider may compute the sum of secret inputs of all other data providers, by subtracting their own input from the resulting sum. When a subset of data providers collude, inferring information about the inputs of other data providers from the output becomes easier, as the colluding set can compute the sum of inputs of non-colluding data providers. To illustrate this, we assume the worst case scenario where $N - 1$ data providers collude. In such a case, the leakage is deterministic as the colluding set can compute the local maximum depth value of the remaining data provider, by subtracting the sum of their inputs from the total sum computed securely. Similarly, they can know the attribute or node label that the remaining data provider votes for. With a colluding set of size $< N - 1$, the leakage becomes probabilistic, and decreases as the size of the colluding set decreases. Even in the worst case, the underlying data records are not directly revealed. To quantify this leakage, the framework proposed by Zhu and Du [54] may be used. It enables the quantification of privacy risks from disclosing decision tree models. This is done by converting the model into a set of constraints, based on the rules in the decision tree. Then, non-linear programming is used to estimate maximum entropy, which is then used to quantify privacy. This can be applied to the trees trained by PrivaTree. Depending on the dataset and the structure of the tree, one can consider each vote by a data provider to be one rule in the tree, assume that this vote is leaked to the other protocol participants, and quantify the leakage associated with it using the framework in [54].

We note that assuming non-collusion is not unreasonable, as all data providers contribute data to the training protocol, and colluding with others would imply having to share their own secret input values. Non-collusion is assumed in similar works [38, 40], including [42] which deals with the most similar setting.

8.5.3 Malicious Adversaries

The privacy guarantees satisfied under the semi-honest threat model apply even when one or more participants are malicious. However, under the malicious threat model, correctness of the output of additive secret-sharing cannot be guaranteed, and as a result, the training of a correct decision tree model is not guaranteed either. Ensuring correctness under the malicious model is beyond the scope of this work.

9 Conclusion

We have designed, implemented and evaluated a protocol for collaboratively training decision trees, in a setting where

the training dataset is horizontally partitioned across multiple data providers, while ensuring data privacy. Our protocol works on datasets with both continuous and categorical attributes, scales well on large training sets, and allows for practical training of complex decision tree models. It also offers a privacy-preserving solution for collaborative selection of the optimal decision tree depth. We achieve privacy and scalability at the cost of a modest loss in model accuracy compared to the central model, while still ensuring that our collaborative protocol outputs a model that outperforms non-collaborative ones. The aforementioned characteristics of PrivaTree make it suitable for biomedical applications. One of many such applications could be the collaborative training of an interpretable decision tree model on patient health records, scattered across a number of medical institutions, for assistance in disease diagnosis. In such a setting, the dataset size is typically large, and the data itself is extremely sensitive. Therefore, scalability and data privacy are both of utmost importance. When it comes to medical decision-making, model interpretability is key, thus decision tree models are an appropriate choice. Since our protocol yields better accuracy than local training, the trained model would be better at correctly predicting diagnosis of new patients based on their health records, and would thus benefit all medical institutions involved. As future work, it would be interesting to design and evaluate extensions of our protocol to decision tree ensembles (e.g., random forests, boosted trees), which would be useful for improving the trained model's accuracy, at the expense of interpretability.

REFERENCES

[1] S. Raudys and A. K. Jain, "Small sample size effects in statistical pattern recognition: recommendations for practitioners," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 252–264, 1991.

[2] R. Shokri and V. Shmatikov, "Privacy-Preserving Deep Learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*. Denver, Colorado, USA: ACM Press, 2015, pp. 1310–1321.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 54. Florida, USA: PMLR, Apr. 2017, pp. 1273–1282.

[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019.

[5] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surv.*, vol. 54, no. 6, 2021.

[6] S. Chen, D. Xue, G. Chuai, Q. Yang, and Q. Liu, "FL-QSAR: a federated learning-based QSAR prototype for collaborative drug discovery," *Bioinformatics*, vol. 36, no. 22-23, pp. 5492–5498, Apr. 2021.

[7] S. Chen, B. Duan, C. Zhu, C. Tang, S. Wang, Y. Gao, S. Fu, L. Fan, Q. Yang, and Q. Liu, "Privacy-preserving integration of multiple institutional data for single-cell type identification with scPrivacy," *Science China Life Sciences*, Dec. 2022.

[8] R. Ma, Y. Li, C. Li, F. Wan, H. Hu, W. Xu, and J. Zeng, "Secure multiparty computation for privacy-preserving drug discovery," *Bioinformatics*, vol. 36, no. 9, pp. 2872–2880, May 2020.

[9] K. Nandakumar, N. Ratha, S. Pankanti, and S. Halevi, "Towards deep neural network training on encrypted data," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Long Beach, CA, USA: IEEE, 2019, pp. 40–48.

[10] C. Kingsford and S. L. Salzberg, "What are decision trees?" *Nature Biotechnology*, vol. 26, no. 9, pp. 1011–1013, Sep. 2008.

[11] G. Stiglic, S. Kocbek, I. Pernek, and P. Kokol, "Comprehensive Decision Tree Models in Bioinformatics," *PLoS ONE*, vol. 7, no. 3, p. e33812, Mar. 2012.

[12] D. Hao, Q. Qiu, X. Zhou, Y. An, J. Peng, L. Yang, and D. Zheng, "Application of decision tree in determining the importance of surface electrohysterography signal characteristics for recognizing uterine contractions," *Biocybernetics and Biomedical Engineering*, vol. 39, no. 3, pp. 806–813, Jul. 2019.

[13] T. Shaikhina, D. Lowe, S. Daga, D. Briggs, R. Higgins, and N. Khovanova, "Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation," *Biomedical Signal Processing and Control*, vol. 52, pp. 456–462, Jul. 2019.

[14] M. Banerjee, E. Reynolds, H. B. Andersson, and B. K. Nallamothu, "Tree-Based Analysis: A Practical Approach to Create Clinical Decision-Making Tools," *Circulation: Cardiovascular Quality and Outcomes*, vol. 12, no. 5, p. e004879, May 2019.

[15] E. De-La-Hoz-Correa, F. E. Mendoza-Palechor, A. De-La-Hoz-Manotas, R. C. Morales-Ortega, and S. H. Beatriz Adriana, "Obesity Level Estimation Software based on Decision Trees," *Journal of Computer Science*, vol. 15, no. 1, pp. 67–77, Jan. 2019.

[16] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[17] J. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 10 2001.

[18] M. Backes, P. Berrang, M. Bieg, R. Eils, C. Herrmann, M. Humbert, and I. Lehmann, "Identifying Personal DNA Methylation Profiles by Genotype Inference," in *2017 IEEE Symposium on Security and Privacy (SP)*. San Jose, CA, USA: IEEE, May 2017, pp. 957–976.

[19] D. J. Wu, T. Feng, M. Naehrig, and K. Lauter, "Privately Evaluating Decision Trees and Random Forests," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 4, pp. 335–355, Oct. 2016.

[20] I. Damgård, D. Escudero, T. K. Frederiksen, M. Keller, P. Scholl, and N. Volgushev, "New primitives for actively-secure MPC over rings with applications to private machine learning," in *2019 IEEE Symposium on Security and Privacy, SP*. San Francisco, CA, USA: IEEE, 2019, pp. 1102–1120.

[21] M. De Cock, R. Dowsley, C. Horst, R. Katti, A. C. A. Nascimento, W.-S. Poon, and S. Truex, "Efficient and

private scoring of decision trees, support vector machines and logistic regression models based on precomputation," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 2, pp. 217–230, 2019.

[22] J. Bai, X. Song, S. Cui, E.-C. Chang, and G. Russello, "Scalable Private Decision Tree Evaluation with Sublinear Communication," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, May 2022, pp. 843–857, arXiv:2205.01284 [cs]. [Online]. Available: http://arxiv.org/abs/2205.01284

[23] Y. Zheng, H. Duan, C. Wang, R. Wang, and S. Nepal, "Securely and Efficiently Outsourcing Decision Tree Inference," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1841–1855, May 2022.

[24] Y. Qu, M. P. Uddin, C. Gan, Y. Xiang, L. Gao, and J. Yearwood, "Blockchain-enabled federated learning: A survey," *ACM Comput. Surv.*, vol. 55, no. 4, nov 2022.

[25] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Comput. Surv.*, vol. 55, no. 3, feb 2022.

[26] S. Truex, L. Liu, M. E. Gursoy, and L. Yu, "Privacy-preserving inductive learning with decision trees," in *2017 IEEE International Congress on Big Data (BigData Congress)*. Honolulu, HI, USA: IEEE, 2017, pp. 57–64.

[27] S. Chatel, A. Pyrgelis, J. R. Troncoso-Pastoriza, and J.-P. Hubaux, "Sok: Privacy-preserving collaborative tree-based model learning," pp. 182–203, 2021.

[28] J. Li, Y. Tian, Y. Zhu, T. Zhou, J. Li, K. Ding, and J. Li, "A multicenter random forest model for effective prognosis prediction in collaborative clinical research network," *Artificial Intelligence in Medicine*, vol. 103, p. 101814, 2020.

[29] A. Patil and S. Singh, "Differential private random forest," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. Delhi, India: IEEE, 2014, pp. 2623–2630.

[30] S. Rana, S. K. Gupta, and S. Venkatesh, "Differentially private random forest with high utility," in *2015 IEEE International Conference on Data Mining*. Atlantic City, NJ, USA: IEEE, 2015, pp. 955–960.

[31] S. Fletcher and M. Islam, "Differentially private random decision forests using smooth sensitivity," *Expert Systems with Applications*, vol. 78, pp. 16–31, 06 2016.

[32] Y. Liu, Z. Ma, Y. Yang, X. Liu, J. Ma, and K. Ren, "Revfrf: Enabling cross-domain random forest training with revocable federated learning," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2021.

[33] Q. Li, Z. Wu, Z. Wen, and B. He, "Privacy-preserving gradient boosting decision trees," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 784–791, Apr. 2020.

[34] Z. Feng, H. Xiong, C. Song, S. Yang, B. Zhao, L. Wang, Z. Chen, S. Yang, L. Liu, and J. Huan, "Securegbm: Secure multi-party gradient boosting," in *2019 IEEE International Conference on Big Data (Big Data)*. Los Angeles, CA, USA: IEEE, 2019, pp. 1312–1321.

[35] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," in *34th AAAI Conference on Artificial Intelligence (AAAI-20)*, vol. 34. AAAI Press, 2020, pp. 4642–4649.

[36] J. Zhan, "Using Homomorphic Encryption For Privacy-Preserving Collaborative Decision Tree Classification," in *2007 IEEE Symposium on Computational Intelligence and Data Mining*. Honolulu, HI, USA: IEEE, 2007, pp. 637–645.

[37] J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson, "Privacy-preserving decision trees over vertically partitioned data," *ACM Transactions on Knowledge Discovery from Data*, vol. 2, no. 3, pp. 1–27, Oct. 2008.

[38] W. Du and Z. Zhan, "Building Decision Tree Classifier on Private Data," in *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining*, ser. CRPIT. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2002, pp. 1–8.

[39] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A Hybrid Approach to Privacy-Preserving Federated Learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security - AISec'19*. London, United Kingdom: ACM Press, 2019, pp. 1–11.

[40] S. de Hoogh, B. Schoenmakers, P. Chen, and H. op den Akker, "Practical secure decision tree learning in a teletreatment application," in *Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, vol. 8437, pp. 179–194.

[41] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, p. 612–613, 1979.

[42] M. Abspoel, D. Escudero, and N. Volgushev, "Secure training of decision trees with continuous attributes," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 1, pp. 167–187, 2020.

[43] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar. 1986. [Online]. Available: https://doi.org/10.1007/BF00116251

[44] ——, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[45] L. Breiman, J. H. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*, ser. The Wadsworth and Brooks-Cole statistics-probability series. Monterey, CA: Taylor & Francis, 1984.

[46] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[47] J. C. Benaloh, "Secret sharing homomorphisms: Keeping shares of a secret sharing," in *Advances in Cryptology - CRYPTO '86*, ser. Lecture Notes in Computer Science, vol. 263. Santa Barbara, California: Springer, 1986, pp. 251–260.

[48] B. Chor and E. Kushilevitz, "A communication-privacy tradeoff for modular addition," *Information Processing Letters*, vol. 45, no. 4, pp. 205–210, 1993.

[49] S. M. Ghanem and I. A. Moursy, "Secure multiparty computation via homomorphic encryption library," in *2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2019, pp. 227–232.

[50] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not,"