SecureBoost: A Lossless Federated Learning Framework

Kewei Cheng[1], Tao Fan[2], Yilun Jin[3], Yang Liu[2], Tianjian Chen[2], Qiang Yang[4]
1. University of California, Los Angeles, Los Angeles, USA
2. Webank, Shenzhen, China
3. Peking University, Beijing, China
4. Hong Kong Unversity of Science and Technology, Hong Kong
keweicheng@g.ucla.edu, dylanfan@webank.com, yljin@pku.edu.cn, yangliu@webank.com
tobychen@webank.com, qyang@cse.ust.hk

Abstract

The protection of user privacy is an important concern in machine learning, as evidenced by the rolling out of the General Data Protection Regulation (GDPR) in the European Union (EU) in May 2018. The GDPR is designed to give users more control over their personal data, which motivates us to explore machine learning frameworks with data sharing without violating user privacy. To meet this goal, in this paper, we propose a novel lossless privacy-preserving tree-boosting system known as SecureBoost in the setting of federated learning. This federated-learning system allows a learning process to be jointly conducted over multiple parties with partially common user samples but different feature sets, which corresponds to a vertically partitioned virtual data set. An advantage of SecureBoost is that it provides the same level of accuracy as the non privacy-preserving approach while at the same time, reveal no information of each private data provider. We theoretically prove that the SecureBoost framework is as accurate as other non-federated gradient tree-boosting algorithms that bring the data into one place. In addition, along with a proof of security, we discuss what would be required to make the protocols completely secure.

Introduction

The modern society is increasingly concerned with the unlawful use and exploitation of our personal data. At the individual level, improper use of personal data may cause potential risk to user privacy. At the enterprise level, data leakage may impinge have grave consequences on commercial interests. Actions are being taken by different societies. For example, the European Union has recently enacted a low known as General Data Protection Regulation (GDPR). The GDPR is designed to give users more control over their personal data (Regulation 2016; Albrecht 2016; Mayer-Schonberger and Padova 2015; Goodman and Flaxman 2016). Many enterprises that rely heavily on machine learning are beginning to make sweeping changes as a consequence.

Despite difficulty in meeting the goal of user privacy protection, the need for different organizations to collaborate while building machine-learning models still stay strong. In reality, many data owners do not have a sufficient amount of data to build high-quality models. For example, retail companies have user transactions data, which correspond to different data dimensions or features as credit-rating companies do. Likewise, mobile phone users have their usage data, but each device only have a small amount of user-activity data. To have a usable model for user preference prediction, it would be necessary to integrate the data collected by the clients.

Thus, it is a challenge to both allow different data owners to collaborate together in order to build high-quality machine learning models while at the same time, protect user data privacy and confidentiality. In the past, several attempts have been made to address the user-privacy problem while exchanging data (Hardy et al. 2017; Mohassel and Zhang 2017). For example, Apple proposed to use *differential privacy* (Dwork, Roth, and others 2014; Dwork 2008) to address the privacy preservation issue. The basic idea of differential privacy (DP) is to add properly calibrated noise to data to disambiguate the identity of any individuals when the data is being exchanged and analyzed by a third party. However, as we discuss in the paper, DP only prevent user-data leakage to a certain degree and cannot completely rule out the identity of an individual. In addition, data exchange under the DP still requires that the data changes hands between organizations, which may not be allowed by strict laws like GDPR. Furthermore, the DP method is *lossy* in machine learning in that models built after noise is injected can reduce much performance in prediction accuracy.

More recently, Google introduces a federated learning framework (Konečnỳ et al. 2016) on its Android cloud. The basic idea is to allow individual clients to encrypt their models which are then uploaded and aggregated at a central cloud site. The machine-learning process at that site can make use of these encrypted models while not leaking the clients' information. This framework applies to a data-partition framework where each partition corresponds to a subset of data samples collected from one or more users.

In this paper, we consider a general setting of multiple parties collaboratively build their machine-learning models while protecting user privacy and data confidentiality. Our setting is as shown in Figure 2. We consider a collection of parties each holding a part of its own data. We can visualize the data located at different parties as a subsection of a big data table that is obtained by taking the union of all data at different parties. Then the data at each party has the following property:

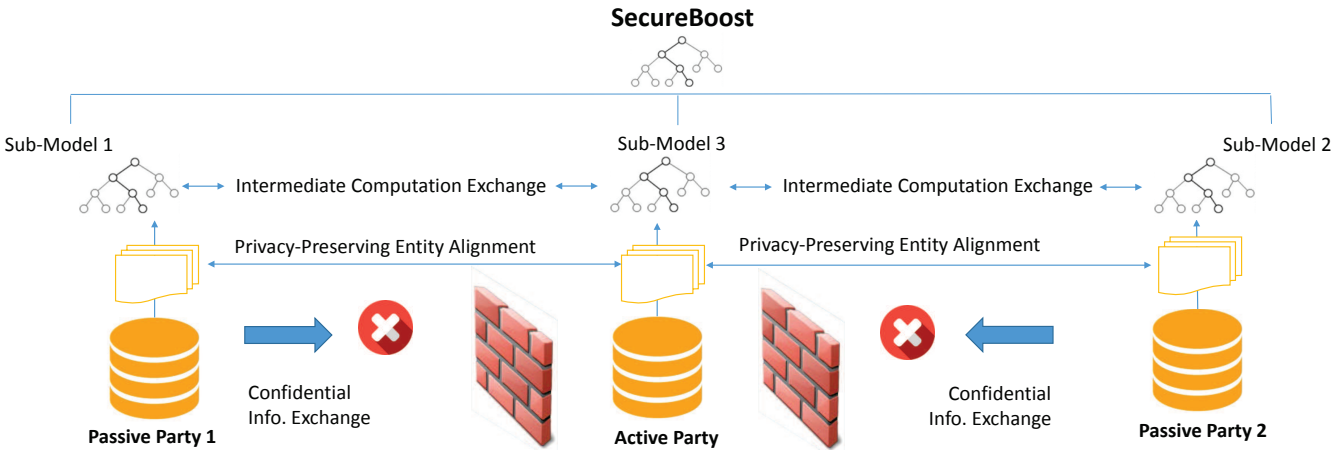


Figure 1: Illustration of the proposed SecureBoost framework

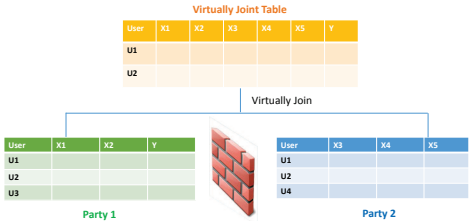


Figure 2: Vertically partitioned data set

1. The big data table is vertically split, such that the data are split in the feature dimension among parties;
2. only one data providers has the label information;
3. the users have partial overlap across different parties.

Our goal is then to allow each party to build a prediction model for some designated label, while disallow any party to obtain any information on the data of other parties.

Our above setting have several advantages. In contrast with most existing work on privacy-preserving data mining and machine learning, the complexity in our setting is significantly increased. Unlike the situation where the data are horizontally split, the above setting requires a more complex mechanism to decompose the loss function at each party (Vaidya 2008; Vaidya and Clifton 2005; Hardy et al. 2017). In addition, in each model-building process for all parties, only one data provider owns the label information. It requires us to propose a secure protocol to guide the learning process instead of sharing label information explicitly among all parties. Finally, data confidentiality and privacy concerns prevents the parties to expose their own users who are not common among the group when building the models. Hence, entity alignment should also be conducted in a sufficiently secure manner.

In this paper, we propose a novel end-to-end privacy-preserving tree-boosting algorithm and framework known as SecureBoost to enable machine learning in a federated setting. Unlike previous federated learning frameworks that split the data on user dimensions, our framework ensures that collaborative model building is done when data is split among different parties on the feature dimension. Our federated learning framework operates in two steps. First, we find the common users among the parties under a privacy-preserving constraint. Then, we collaboratively learn a shared classification or regression model without leaking any user information to each other. We summarize our main contributions as follows:

- We formally define a novel problem of privacy-preserving machine learning over vertically partitioned data in the setting of federated learning.
- We present an approach to train a high-quality tree boosting model for each party collaboratively while keeping the training data secret over multiple parties. We go through this machine learning process without the participation of a trusted third party.
- Finally and importantly, we prove that our approach is *lossless* in the sense that it is as accurate as any centralized non-privacy-preserving methods that bring all data to a central location.
- In addition, along with a proof of security, we discuss what would be required to make the protocols completely secure.

Preliminaries and Related Work

The existing literature on privacy-preserving machine learning broadly address two objectives: privacy of the data used for learning a model or as input to an existing model. To protect privacy of the data used for learning a model, in (Shokri and Shmatikov 2015; Abadi et al. 2016), the authors propose to take advantage of differential privacy for learning a deep learning model. As one of the most popular privacy-preserving techniques, differential privacy (Dwork 2008) protects sensitive data by injecting noise to the raw datasets such that the amount of information leaked from an individual record is minimized. Even though

differential privacy ensures a pretty low probability of identifying an individual record, there's still a probability of leakage, which is against the requirement of GDPR. To address the above problems, Google introduces a federated learning framework to bring the model training to each mobile terminal (Konečnỳ et al. 2016). It achieves the goal of privacy protection by forbidding the data from transferring out. Another privacy preserving techniques is focuses on the inference stage instead of training stage. Microsoft proposed a cryptographic deep learning framework, CryptoNets (Gilad-Bachrach et al. 2016) based on Homomorphic Encryption to enable a trained neural network to make encrypted predictions over the encrypted data. However, it has to sacrifice the accuracy to obtain security. In (Rouhani, Riazi, and Koushanfar 2017), another framework DeepSecure is proposed to securely conduct deep learning execution on encrypted data using Yao's Garbled Circuit (GC) protocol. Although it does not involve a tradeoff between utility and privacy, it suffers from serious inefficiency.

All the above methods are designed for horizontally partitioned data whose data providers record the same features for different entities. We consider a vertical data partition as shown in Figure 2, in which multiple parties record different features at different sites. Different from the horizontal partitioning, which assumes that ensemble happens over data samples, the vertical partition builds a model over a common set of users. How to collaboratively build the model is an open question. Some previous works discuss privacy-preserving decision trees over vertically partitioned data (Vaidya and Clifton 2005; Vaidya et al. 2008). However, their proposed methods have to reveal class distribution over the given attributes, which will cause potential security risk. In addition, they can only handle discrete data, which is less practical for real-life scenario. In contrast, our method guarantees more secure protection to the data and can easily apply to continuous data. In (Djatmiko et al. 2017), Patrini et al. proposed a framework to jointly perform logistic regression over the encrypted vertically-partitioned data by approximating a non-linear logistic loss by a Taylor expansion. Clearly, in this approximation, the algorithm will inevitably cause a loss of accuracy. To the contrary, we propose a novel approach that is *lossless* in nature. We believe that the SecureBoost framework is the first attempt for privacy-preserving federated learning over vertically partitioned data which balance accuracy and security.

Problem Statement

We now formally define our problem and clarify the difference between our setting and previous works. Let $\left\{\mathbf{X}^k \in \mathbb{R}^{n_k \times d_k}\right\}_{k=1}^m$ be the data matrix distributed on m private parties with each row $\mathbf{X}_{i*}^k \in \mathbb{R}^{1 \times d_k}$ being a data instance. We use $\mathcal{F}^k = \{f_1, ..., f_{d_k}\}$ to denote the feature set of corresponding data matrix $\mathbf{X}^k$. If we consider all data come from a virtual big data table involving all users and all features, then we can view the data as being vertically split from a large virtual table across different parties, such that each party holds a different set of vertically partitioned data over a subset of users. Two parties p and q have different sets of features, denoted as $\mathcal{F}^p \cap \mathcal{F}^q = \varnothing, \forall p \neq q \in \{1...m\}$. Different data providers may hold different sets of users as well, allowing some degree of overlap. That is, parties at sites $n_1...n_m$ may be different from each other. As mentioned before, when building a model for a common task, we consider that only one of the data providers has a class attribute for classification or regression. We denote the class label as $\mathbf{y} \in \mathbb{R}^{n_k \times 1}$ where the class label is held by the k-th party.

Definition 1. ***Active Party:***

We define the active party as the data provider who holds both a data matrix and the class label.

Since the class label information is indispensable for supervised learning, there must be an active party with access to the label $\mathbf{y}$. The active party naturally takes the responsibility as a dominating *server* in federated learning.

Definition 2. ***Passive Party:***

We define the data provider which has only a data matrix as a passive party.

Passive parties play the role of clients in the federated learning setting. They are also in need of building a model to predict the class label $\mathbf{y}$ for their prediction purposes. Thus they must collaborate with the active party to build their model to predict $\mathbf{y}$ for their future users using their own features.

The problem of privacy-preserving machine learning over vertically partitioned data in federated learning can be stated as follows:

Given: a vertically partitioned data matrix $\left\{\mathbf{X}^k\right\}_{k=1}^m$ distributed on m private parties and the class labels $\mathbf{y}$ distributed on active party.

Learn: a machine learning model M without giving information of the data matrix of any parties to others in the process. The model M is a function that has a projection M_i at each party i, such that M_i takes input of its own features X_i.

Lossless Constraint: We require that the model M is lossless, which means that the loss of M under federated learning over the training data is the same as the loss of M' when M' is built on the union of all data.

Federated Learning with SecureBoost

As one of the most widely-used machine-learning algorithms, the gradient-tree boosting model (Friedman et al. 2000) excels in many machine learning tasks, such as fraud detection (Oentaryo et al. 2014), feature selection (Li et al. 2017) and product recommendation (He et al. 2014). In this section, we propose a novel gradient-tree boosting algorithm we call SecureBoost in the setting of federated learning. As shown in Figure 1, SecureBoost consists of two major steps. First, it aligns the data under the privacy constraint. Second, it collaboratively learn a shared gradient-tree boosting model while keeping all the training data secret over multiple private parties. Below, we explain each part in turn.

Our first goal is to find a common set of data samples at all participating parties so as to build a joint model M. When the data is vertically partitioned over multiple parties, different parties hold different but partially overlapped users. These users may be identified by their unique user IDs. A problem is how to find the common shared users or data samples across the parties without compromising the non-shared parts of the user sets. In particular, we align the data samples under an encryption scheme by using the privacy-preserving protocol for inter-database intersections (Liang and Chawathe 2004).

After aligning the data across different parties under the privacy constraint, we now consider the problem of jointly building tree ensemble model over multiple parties without violating privacy in federated learning. Before further discussing the detail of the algorithm, we first introduce a general framework of federated learning. In federated learning, a typical interation consists of four steps. At first, each client downloads the current global model from server. Next, each client computes an updated model based on its local data and the current global model, which resides with the active party. Third, each client sends the model update back to the server under encryption. Finally, the server aggregates these model updates and construct the improved global model.

Following the general framework of federated learning, we can see that to achieve a privacy-preserving tree boosting framework in the setting of federated learning, in essence, we have to answer the following three questions: (1) How can each client (i.e., a passive party) compute an updated model based on its local data without reference to class label? (2) How can the server (i.e., the active party) aggregate all the updated model and obtain a new global model? (3) How to share the updated global model among all parties without leaking any information at inference time? To answer these three questions, we start by reviewing a tree ensemble model, XGBoost (Chen and Guestrin 2016), in a non-federated setting.

Given a data set $\mathbf{X} \in \mathbb{R}^{n \times d}$ with n samples and d features, XGBoost predicts the output by using K regression trees.

$$\hat{y_i} = \sum_{k=1}^{K} f_k(\mathbf{x}_i) \tag{1}$$

To learn the set of regression tree models used in Eq.(1), it greedily adds a tree f_t at the t-th iteration to minimize the following loss.

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y_i}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t) \tag{2}$$

where $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \left\|w\right\|^2$, $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$.

When construct the regression tree in the t-th iteration, it starts from the tree with the depth of 0 and add a split for each leaf nodes of the tree until reaching the maximum depth. In particular, it employs the following equation to determine the best split.

$$\mathcal{L}_{split} = \frac{1}{2}[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda}] - \gamma \tag{3}$$

In the above equation, I_L and I_R are the instance spaces of left and right tree nodes after the split. The split that maximizes the score is selected as the best split.

When it obtains an optimal tree structure, the optimal weight w_j^* of leaf j can be computed by the following equation:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{4}$$

where I_j is the instance space of leaf j.

From the above review, we make following observations:

(1) The evaluation of split candidates and the calculation of the optimal weight of leaf only depend on the g_i and h_i.

(2) The class label is needed for the calculation of g_i and h_i. For instance, when we take the logistic loss as the loss function, we have $g_i = -y_i(1 - \frac{1}{1+e^{-y_i^{(t-1)}}}) + (1 - y_i)\frac{1}{1+e^{-y_i^{(t-1)}}}$ and $h_i = \frac{e^{-y_i^{(t-1)}}}{(1+e^{-y_i^{(t-1)}})^2}$. Hence, it is easy to recover the class label from g_i and h_i once we obtain the value of $y_i^{(t-1)}$.

With the guidance of the above observations, we now discuss how to adapt a non-federated gradient boosted tree model to a federated learning setting. Following observation (1), we can see that each passive party can determine the locally optimal split independently with only its local data once it obtains g_i and h_i. Thus, a naive solution is requiring the active party to send g_i and h_i to each passive party. However, according to observation (2), g_i and h_i should be regarded as sensitive data as well, since they can be used to discover the class label information. To ensure security, passive parties cannot get access to g_i and h_i directly. In order to keep g_i and h_i confidential, we require the active party to encrypt g_i and h_i before sending them to passive parties. The remaining challenge is how to determine the locally optimal split with encrypted g_i and h_i for each passive party.

According to Eq.(5), the optimal split can be found if $g_l = \sum_{i \in I_L} g_i$ and $h_l = \sum_{i \in I_L} h_i$ can be calculated for every possible splits, where I_L is the instance space of left nodes after the split. Next, we show how to obtain g_l and h_l with encrypted g_i and h_i using additively homomorphic encryption scheme (Paillier 1999).

First, we define the encryption of a number u under additively homomorphic encryption scheme as $\langle u \rangle$. Recalling the main properties of additively homomorphic encryption scheme, for any two numbers u and v, we have $\langle u \rangle + \langle v \rangle = \langle u + v \rangle$. Therefore, $\langle h_l \rangle$ is equivalent to $\sum_{i \in I_L} \langle h_i \rangle$ and similarly, $\langle g_l \rangle$ can be computed by $\sum_{i \in I_L} \langle g_i \rangle$. By taking advantage of additively homomorphic encryption scheme, the best split can be found in the following way. First, each passive party computes $\langle g_l \rangle$ and $\langle h_l \rangle$ for all possible splits locally. It then sends the values back to the active party. After collecting the values from all passive parties, the active party deciphers $\langle g_l \rangle$ and $\langle h_l \rangle$ and calculates the global op-

Algorithm 1 Aggregate Encrypted Gradient Statistics

Input: I, instance space of current node
Input: d, feature dimension
Input: $\{\langle g_i \rangle, \langle h_i \rangle\}_{i \in I}$
Output: $\mathbf{G} \in \mathbb{R}^{d \times l}$, $\mathbf{H} \in \mathbb{R}^{d \times l}$

for $k = 0 \rightarrow d$ **do**
  Propose $S_k = \{s_{k1}, s_{k2}, ..., s_{kl}\}$ by percentiles on feature k
end for
for $k = 0 \rightarrow d$ **do**
  $\mathbf{G}_{kv} = \sum_{i \in \{i | s_{k,v} \geq x_{i,k} > s_{k,v-1}\}} \langle g_i \rangle$
  $\mathbf{H}_{kv} = \sum_{i \in \{i | s_{k,v} \geq x_{i,k} > s_{k,v-1}\}} \langle h_i \rangle$
end for

timal split according to Eq.(5). In this case, the communication cost between the active and each passive parties is $2 * n * d * ct$ for a single split. Here, ct denotes the size of ciphertext, n represents the number of instances associated with the node to be split and d is the number of features held by the passive party.

We can observe that this solution is not efficient since it requires the transfer of $\langle g_l \rangle$ and $\langle h_l \rangle$ for all possible split candidates. To construct the tree with lower communication cost, we take advantage of an approximate framework proposed by (Chen and Guestrin 2016), where the detailed calculation is shown in Algorithm 1. For each passive party, instead of computing $\langle g_l \rangle$ and $\langle h_l \rangle$ directly, it maps the features into buckets and then aggregates the encrypted gradient statistics based on the buckets. In this way, the active party only needs to collect the aggregated encrypted gradient statistics from all passive parties. As a result, it can determine the globally optimal split as described in Algorithm 2. In this case, the communication cost for constructing a regression tree can be reduced to $2 * (n/q) * d * ct$ where q denotes the number of instances in one bucket. Clearly, we have $(1/q) << 1$. Therefore, we can indeed decrease the communication cost. After the active party obtains the global optimal split, [party id (i), feature id (k), threshold id (v)], it returns the feature id k and threshold id v to the corresponding passive party i. Passive party i decides the selected attribute's value based on the value of k and v. Then, it partitions the current instance space according to the selected attribute's value. In addition, it builds a lookup table locally to record the selected attribute's value, [feature, threshold value], as shown in Figure 3. After that, it returns the index of the record and the instance space of left nodes after the split (I_L) back to the active party. The active party splits the current node according to the received instance space and associate current node with [party id, record id], until a stopping criterion or the max depth is reached. All the leaf nodes are stored inside the active party.

Federated Inference based on the Learned Model

In this section, we describe how to use the learned model (distributed among parties) to classify a new instance even though the features of the instance to be classified are private and distributed among parties. Since each site knows the its own features (and can thus evaluate the branch), but knows nothing of the others, we need a secure distributed protocol to control passes from site to site, based on the decision made.

To illustrate the inference process, we consider a a system with three parties as depicted in Figure 3. Specifically, party 1 is the active party, which collects information about user's monthly bill payment and level of education, as well as the label information. Party 2 and party 3 are passive parties, which hold the features, [age, gender,marriage status] and [amount of given credit] respectively. Suppose we wish to know if a user $X6$ would make payment on time. All sites have to collaborate to make the prediction. The whole process is coordinated by the active party. Starting from the root, by referring to the record [party id:1, record id:1], the active party knows party 1 holds the root node. Thereby, it requires party 1 to retrieve the corresponding attribute, Bill Payment, from its lookup table based on the record id 1. Since the classifying attribute is bill payment and party 1 knows the bill payment for user $X6$ is 4367, which is less than the threshold value 5000, it makes the decision that it should move down to its left child, node 1. Then, active party refers to the record [party id:3, record id:1] associated with node 1 and requires party 3 to conduct the same operations.

Algorithm 2 Split Finding

Input: I, instance space of current node
Input: $\{\mathbf{G}^i, \mathbf{H}^i\}_{i=1}^m$, aggregated encrypted gradient statistics from m parties
Output: Partition current instance space according to the selected attribute's value

/*Conduct on Active Party*/
$g \leftarrow \sum_{i \in I} g_i, h \leftarrow \sum_{i \in I} h_i$
//enumerate all parties
for $i = 0 \rightarrow m$ **do**
  //enumerate all features
  for $k = 0 \rightarrow d_i$ **do**
    $g_l \leftarrow 0, h_l \leftarrow 0$
    //enumerate all threshold value
    for $v = 0 \rightarrow l_k$ **do**
      get decrypted values $D(\mathbf{G}^i_{kv})$ and $D(\mathbf{H}^i_{kv})$
      $g_l \leftarrow g_l + D(\mathbf{G}^i_{kv}), h_l \leftarrow h_l + D(\mathbf{H}^i_{kv})$
      $g_r \leftarrow g - g_l, h_r \leftarrow h - h_l$
      $score \leftarrow max(score, \frac{g_l^2}{h_l + \lambda} + \frac{g_r^2}{h_r + \lambda} - \frac{g^2}{h + \lambda})$
    end for
  end for
end for
  return k_{opt} and v_{opt} to the corresponding passive party i when we obtain the max score.
  /*Conduct on Passive Party i*/
  determine the selected attribute's value according to k_{opt} and v_{opt} and partition current instance space.
  record the selected attribute's value and return [record id, I_L] back to the active party.
  /*Conduct on Active Party*/
  split current node according to I_L and associate current node with [party id, record id].

This process continues until a leaf is reached.

Theoretical Assessment for Lossless Property

Theorem 1. *SecureBoost is lossless as defined in Section Problem Statement, provided that the model M and M' have the same initialization and parameters.*

Proof. The loss of the model M under the setting of federated learning is the same as the loss of M' when M' is built on the union of all data, because M' and M' are identical. According to Eq.(5), g_i and h_i is the only information needed for the calculation of best split. Provided that with the same initialization, each iteration, each instance has the same value of g_i and h_i under both settings, model M and M' can always achieve the same best split during the construction of the tree. Thereby, M' and M' are identical, which ensures the property of lossless. □

Security Discussion

In this section, we discuss the security of our proposed SecureBoost framework. In particular, we will provide detailed analysis of information leakage of the framework and discuss the security of our framework in the presence of semi-honest adversaries. In addition, along with a proof of security, we discuss what would be required to make the protocols completely secure.

Analysis of Information Leakage

As SecureBoost consists of two components, we discussion information leakage of these two component respectively.

During privacy-preserving entity alignment, the encryption techniques guarantees that nothing reveals but the ID of the common shared users across the parties. Although revealing ID of the common shared users might cause some potential risk, these level of leakage is acceptable in most scenarios.

For the construction of the tree ensemble model, all that is revealed contains: (1) Each party knows instance space for the each split; (2) Each party knows the tree nodes held by itself; (3) Active party knows the number of features held by each passive parties; (4) Active party knows the actual value of g_l and h_l; (5) Active party knows which site is responsible for the decision made at each node. Considering a system with one passive party and one active party, we now discuss the potential security risk caused by the leaked information.

First, we study how much information the passive parties can learn about active party. As we know, SecureBoost essentially is a decision tree model. Although its leaf nodes do not hold a class label, instances associated with the same leaf still strongly indicates that they may belong to the same class or result in similar regression results. Thereby, in SecureBoost, we require leaf nodes to be unknown to passive party in order to prevent the label information from disclosure. However, such protection is not enough to guarantee the security. Let us consider the situation that a passive party holds the parent node of two leaf nodes. In this case, the instances space of those leaf nodes is no longer hidden from passive party. Passive party can make a guess that all instances associated with the same leaf belong to the same class. The confidence of the inference is determined by leaf purity where leaf purity refers to the proportion of samples which belong to the majority class. Thus, we take leaf purity as metrics to give a quantitative information leakage analysis to SecureBoost. More precisely, we consider the scenario of binary classification for the reason that it will potentially cause the greatest security risk.

According to Eq.(2), to learn the SecureBoost model, we greedily add a decision tree f_t at the t-th iteration to fitting residual $y_i - \hat{y_i}^{(t-1)}$. Therefore, when $t > 1$, the instances associated with the same leaf only indicate that they may have similar residual, which cannot directly used to infer the label information. However, when $t = 1$, f_1 try to fit the label y_i. In this case, the instance space of the leaf nodes may reveal the label information. Thereby, our security concern mainly focus on how much information we can infer from the first tree, f_1. Let's start our analysis with Theorem 2.

Theorem 2. *For a learned SecureBoost model, the information leakage is given by the weight of the first tree's leaves.*

Proof. The loss function for binary classification problem is shown as follows.

$$L = y_i log(1 + e^{-\hat{y}_i}) + (1 - y_i) log(1 + e^{\hat{y}_i}) \qquad (5)$$

Based on the loss function, we have $g_i = \hat{y_i}^{(0)} - y_i$ and $h_i = \hat{y_i}^{(0)} * (1 - \hat{y_i}^{(0)})$ during the construction of the decision tree at first iteration. Specifically, $\hat{y_i}^{(0)}$ is given as initialized value. Suppose we initialize all $\hat{y_i}^{(0)}$ as a where $0 < a < 1$. According to Eq.(4), for the instances associated with the specific leaf j, $\hat{y_i}^{(1)} = S(w_j^*) = S(-\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda})$ where $S(x)$ is the sigmoid function. Suppose the number of instances associated with the leaf j is n_j and the percentage of positive samples is θ_j. When n_j is relatively big, we can ignore λ. Therefore, we have $w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i} = -\frac{\theta * n * (a-1) + (1-\theta) * n * a}{n * a * (1-a)} = -\frac{\theta * n * (a-1) + (1-\theta) * n * a}{n * a * (1-a)} = \frac{a-\theta}{a(a-1)}$

Notice $\max(\theta, 1-\theta)$ is the leaf purify of leaf j. In another word, given a learned SecureBoost model, the information leakage can be inferred from the weight of the first tree's leaves.

□

According to Theorem 2, as long as weight of the first tree's leaves are close enough to $S(\frac{2a-1}{2a(a-1)})$, the protocol is considered secure.

Second, we focus on whether the active party can learn about private information of passive party. Specifically, we have security concern that if active party can recover portion of features held by passive parties with some confidence. During training, active party learns (1) instance space for the each split; (2) tree nodes held by itself; (3) the number of features held by each passive parties; (4) the actual value of g_l and h_l; (5) which site is responsible for the decision made at each node. To recover the features, active party has to learn partial order relation among all instances regarding to a specific feature. However, the only information it knows

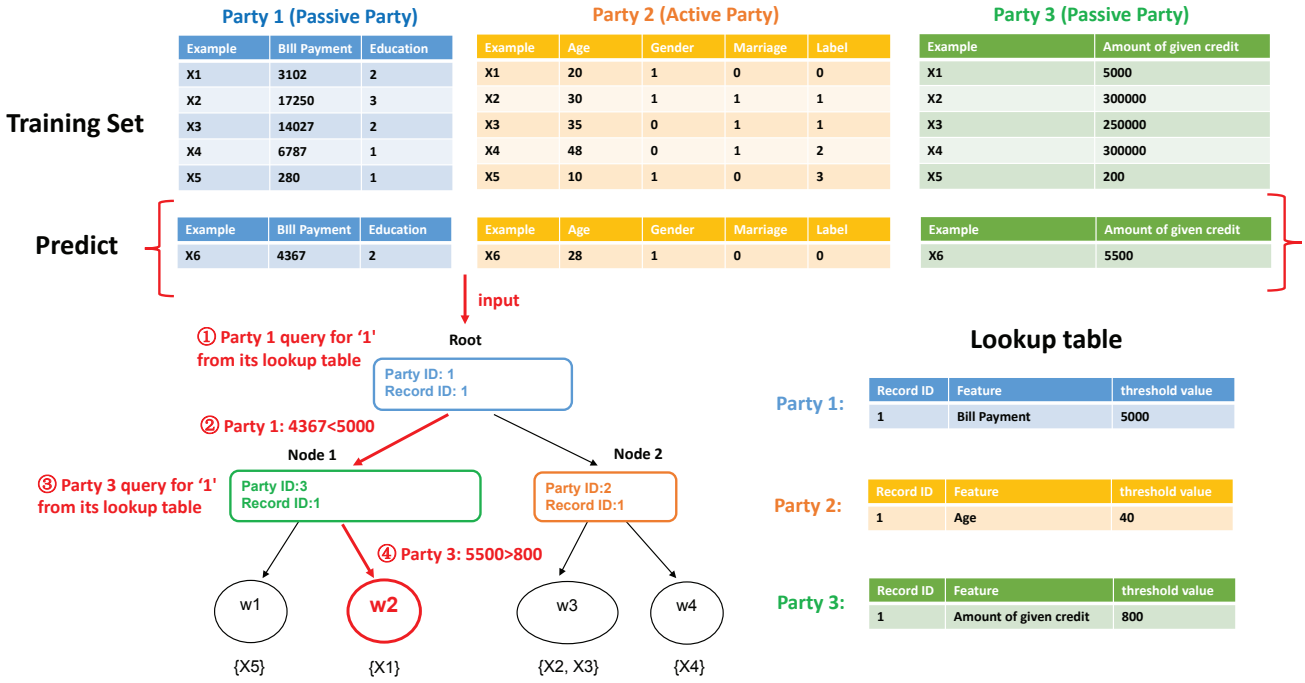


Figure 3: An illustration of prediction.

is how best splits partition the instance space, which is obvious not enough to learn the partial order relation.

Generally speaking, the level of information leakage for SecureBoost is acceptable based on our analysis.

Semi-Honest Security

In this subsection, we would like to discuss security of our framework under the semi-honest assumption. In our security definition, all parties are honest-but-curious. Some corrupt parties might cooperate with each other in order to gather private information. Specifically, we require active party does not collude with any passive party. We now prove that Secureboost is secure under the security definition.

Proof. Our SecureBoost system can be split into two parts, the first part includes only active party and the second part includes all passive parties. When all passive parties collude, the system is equal to a system with one active party and one super passive party. This super passive party holds all feature from passive parties. As discussed in Section Analysis of Information Leakage, we have proved that when our system has only one active party and one passive party, the level of information leakage is acceptable. Therefore, our system is secure under the semi-honest assumption. □

Completely SecureBoost

As discussed in Section Analysis of Information Leakage, our main security concern is that instance space for leaf nodes may reveal too much information and the passive party indeed has chance to know instance space for the leaf nodes when collaboratively construct the tree ensemble model with the active party. To alleviate this problem, we proposed *Completely SecureBoost* to prevent the passive party from constructing the first tree. Unlike *SecureBoost*, the active party of *Completely SecureBoost* learns the first tree independently based on its own features, rather than collaborate with passive parties. Thereby, the instance space of the leaf nodes of the first tree can be protected. In this case, all that passive party can learn is the residuals. Although we intuitively illustrate that residuals won't reveal much information once the first tree get protected, to make it more plausible, we now give a theoretical proof as presented in Theorem 3.

Theorem 3. *The residuals of the tree won't reveal much information when leaf purity of the previous tree is high.*

Proof. As mentioned before, for binary classification problem, we have $g_i = \hat{y_i}^{(t-1)} - y_i$ and $h_i = \hat{y_i}^{(t-1)} * (1 - \hat{y_i}^{(t-1)})$, where $g_i \in [-1, 1]$. Hence,

$$\begin{aligned} &\text{if } y_i = 0, h_i = g_i(1 - g_i) \\ &\text{if } y_i = 1, h_i = -g_i(g_i + 1) \end{aligned} \tag{6}$$

When we construct the decision tree at the t-th iteration with k leaves to fit the residuals of the previous tree, in essential, we split the data into k clusters to minimize the following loss.

$$
\begin{aligned}
L &= -\sum_{j=1}^{k} \frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i} \\
&= -\sum_{j=1}^{k} \frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j^N} g_i(1-g_i) + \sum_{i\in I_j^P} -g_i(1+g_i)}
\end{aligned} \tag{7}
$$

We know $\hat{y_i}^{(t-1)} \in [0,1]$ and $g_i = \hat{y_i}^{(t-1)} - y_i$. Thus, we have $g_i \in [-1,0]$ for positive samples and $g_i \in [0,1]$ for negative samples. Taking the range of g_i into consideration, we rewrite the above equation as follows.

$$
\sum_{j=1}^{k} \frac{(\sum_{i\in I_j^N} |g_i| - \sum_{i\in I_j^P} |g_i|)^2}{\sum_{i\in I_j^N} |g_i|(|g_i|-1) + \sum_{i\in I_j^P} |g_i|(|g_i|-1)} \tag{8}
$$

Where I_j^N and I_j^P denote the set of negative samples and positive samples associated with leaf j respectively. We denote the expectation of $|g_i|$ for positive samples as μ_p and the expectation of $|g_i|$ for negative samples as μ_n. When we have a large amount of samples but small number of leave nodes k, we can use the following equation to approximates Eq.(8).

$$
\sum_{j=1}^{k} \frac{(n_j^n \mu_n - n_j^p \mu_p)^2}{n_j^n \mu_n(\mu_n - 1) + n_j^p \mu_p(\mu_p - 1)} \tag{9}
$$

Where n_j^n and n_j^p represent the number of negative samples and positive samples associated with leaf j. Since $\mu_n \in [0,1]$ and $\mu_n \in [0,1]$, we know the numerator has to be positive and the denominator has to be negative. Thus, the whole equation has to be negative. To minimize Eq.(9) is equal to maximizing the numerator while minimizing the denominator. Notice that the denominator is $\sum x^2$ and the numerator is $(\sum x)^2$ where $x \in [0,1]$. The equation is dominated by numerator. Thereby, minimizing Eq.(9) can be regarded as maximizing the numerator $(n_j^n \mu_n - n_j^p \mu_p)^2$. Ideally, we require $n_j^n = n_j^p$ in order to prevent label information from divulging. When $|\mu_n - \mu_p|$ is bigger, more possible we can achieve the goal. And we know $|g_i| = |\hat{y_i}^{(t-1)} - y_i| = \hat{y_i}^{(t-1)}$ for negative samples and $|g_i| = |\hat{y_i}^{(t-1)} - y_i| = 1 - \hat{y_i}^{(t-1)}$ for positive samples. Thereby, $\mu_n = \frac{1}{N_n}\sum_{j=1}^{k}(1-\theta_j)n_j\hat{y_i}^{(t-1)}$ and $\mu_p = \frac{1}{N_p}\sum_{j=1}^{k}\theta_j n_j(1-\hat{y_i}^{(t-1)})$. $|\mu_n - \mu_p|$ can be calculated as follows.

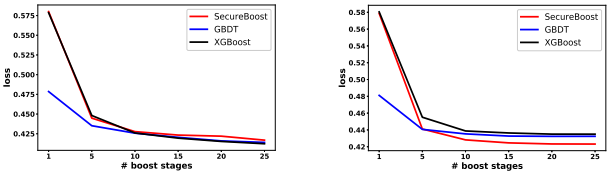
$$
\begin{aligned}
&|\mu_n - \mu_p| \\
=&|\frac{1}{N_n}\sum_{j=1}^{k}(1-\theta_j)n_j\hat{y_i}^{(t-1)} - \frac{1}{N_p}\sum_{j=1}^{k}\theta_j n_j(1-\hat{y_i}^{(t-1)})|
\end{aligned} \tag{10}
$$

Where N_n and N_p correspond to the number of negative samples and positive samples in total. θ_j is the percentage of positive samples associated with leave j for decision tree at $(t-1)$-th iteration (previous decision tree). n_j denote the number of instances associated with leave j for previous decision tree. $\hat{y_i}^{(t-1)} = S(w_j)$ where w_j represents the weight of j-th leave of previous decision tree. When the positive samples and negative samples are balanced, $N_n = N_p$, we have

$$
\begin{aligned}
&|\mu_n - \mu_p| \\
=&\frac{1}{N_n}|\sum_{j=1}^{k}((1-\theta_j)n_j S(w_j) - \theta_j n_j(1-S(w_j)))| \\
=&\frac{1}{N_n}\sum_{j=1}^{k} n_j|(S(w_j) - \theta_j)| \\
=&\frac{1}{N_n}\sum_{j=1}^{k} n_j|(S(\frac{a-\theta_j}{a(a-1)}) - \theta_j)|
\end{aligned} \tag{11}
$$

As observed from Eq.(11), it achieves the minimum value when $S(\frac{a-\theta_j}{a(a-1)}) = a$. By solving the equation, we have the optimal solution of θ_j as $\theta_j* = a(1+(1-a)\ln(\frac{a}{1-a}))$. In order to achieve bigger $\mu_n - \mu_p$, we want the deviation from θ_j to θ_j* to be as big as possible. When we have proper initialization of a, for instance $a = 0.5$, $\theta_j* = 0.5$. In this case, maximizing $|\theta_j - \theta_j*|$ is the same as maximizing $\max(\theta_j, 1-\theta_j)$, which exactly is the leaf purity. Therefore, we have proved that high leaf purity will guarantee big difference between μ_n and μ_p, which finally results in less information leakage. We complete our proof. □

Given Theorem 3, we can conclude that *Completely SecureBoost* is secure when its first tree learn enough information to mask the actual label with residuals.



(a) Learning Curve (b) Test Error

Figure 4: Loss convergence

Experiments

In this section, we conduct experiments on two public datasets. The summary of these datasets is shown as follows.

Credit 1[1]: It involves the problem of classifying whether a user would suffer from serious financial problems. It contains a total of 150000 instances and 10 attributes.

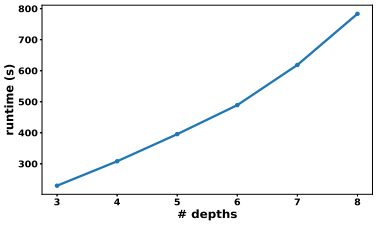
Credit 2[2]: It is also a credit scoring dataset, which is correlated to the task of predicting whether a user would make payment on time. It consist of 30000 instances and 25 attributes in all.

[1]https://www.kaggle.com/c/GiveMeSomeCredit/data

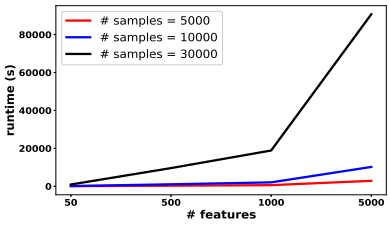
[2]https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset

Table 1: Runtime for Entity Alignment

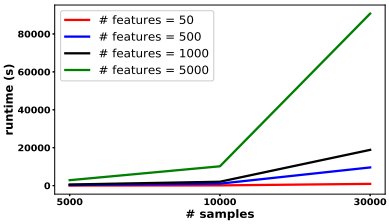
# samples A \ # samples B	$1K$	$10K$	$100K$	$1M$
$1K$	0.758570	3.1834986	47.849709	480.712245
$10K$	5.278931	9.650333	53.428195	484.582641
$100K$	53.032154	54.919945	97.323001	547.061498
$1M$	529.684710	532.306558	584.671372	1021.334617



(a)Runtime w.r.t. maximum depth of individual tree



(b)Runtime w.r.t. feature size



(c)Runtime w.r.t. sample size

Figure 5: Scalability Analysis of Secure Federated Tree Boosting System

In our experiment, we use $2/3$ of the datasets for training and the remained for testing. We split the data vertically into two halves and distribute them to two parties. To fairly compare different methods, we set the maximum depth of individual regression tree as 3, the fraction of samples to be used for fitting the individual regression trees as 0.8 and learning rate as 0.3 for all methods. The Paillier encryption scheme is taken as our additively homomorphic scheme with a key size of 512 bits. All experiments are conducted on a machine with 8GB RAM and Intel Core $i5-7200u$ CPU.

Scalability

As SecureBoost consists of two components, the privacy-preserving entity alignment and the secure federated tree boosting system, we study the scalability of each component separately.

Efficiency of Privacy-Preserving Entity Alignment We consider a system with only two parties when evaluating the scalability of privacy-preserving entity alignment algorithm. The number of samples distributed on parties A and B are important factors to consider. To investigate the effects of two factors, we vary the number of samples distributed on parties A and B on the log-scale from $1K$ to $1M$ separately. We study the effect of each variation by fixing the other to investigate how the change affects the running time. The results are shown in the Table 1 with the following observations.

- In general, the runtime variation w.r.t. the size of samples distributed on party A has a similar trend as the variation of the size of samples distributed on party B, which suggests that the number of samples distributed on party A and B, respectively, contribute equally to running time.
- The running time strongly depends on max(#samples A, #samples B). When the size of samples distributed on party A is equal to the samples distributed on party B, the runtime increases almost linearly with the increase of the size of samples.
- It only takes around 16 minutes of computation time to align entities when the number of samples distributed on both parties A and B are $1M$, which is fairly efficient. This observation validates the scalability of our entity alignment algorithms.

Table 2: First Tree vs. Second Tree in terms of Leaf Purity

Mean Purity	Credit 1	Credit 2
1st tree	0.8058	0.7159
2rd tree	0.66663	0.638

Efficiency of Secure Federated Tree Boosting System We notice that the effectiveness of secure federated tree boosting system may be influenced by (1) convergence rate; (2) maximum depth of the individual regression tree; (3) the sample size of the datasets; and (4) the feature size of the datasets. In this subsection, we study the impact of all four variables on the runtime of learning respectively. All experiments are conducted on dataset Credit 2.

First, we are interested in the convergence rate of our proposed system. We compare the convergence rate of SecureBoost with non-federated tree boosting implementation, including GBDT[3] and XGBoost[4]. As can be observed from the Figure 4, SecureBoost shows a similar learning curve with other non-federated baseline methods on the training dataset. It performs slightly better than others on the test

[3] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[4] https://github.com/dmlc/xgboost

Table 3: SecureBoost vs. Completely SecureBoost in terms of Classification Performance

Accuracy	Credit 1	Credit 2
1st Tree of *SecureBoost*	0.9298	0.7806
1st Tree of *Completely SecureBoost*	0.9186	0.7793
Overall Performance of *SecureBoost*	0.9345	0.8180
Overall Performance of *Completely SecureBoost*	0.9331	0.8179

F1-score	Credit 1	Credit 2
1st Tree of *SecureBoost*	0.012	0
1st Tree of *Completely SecureBoost*	0	0
Overall Performance of *SecureBoost*	0.2576	0.4634
Overall Performance of *Completely SecureBoost*	0.2549	0.4650

AUC	Credit 1	Credit 2
1st Tree of *SecureBoost*	0.7002	0.6381
1st Tree of *Completely SecureBoost*	0.6912	0.6320
Overall Performance of *SecureBoost*	0.8461	0.7701
Overall Performance of *Completely SecureBoost*	0.8423	0.7682

dataset. In addition, we can see that with the increase of boost stages, both training loss and testing loss drop rapidly at first. When the boosting stages keep increasing from 10 to 25, the loss does not vary much on both the training dataset and the test dataset. To sum up, the algorithm performs quite well in terms of convergence, which is appealing in practice as it significantly reduces the computational costs.

Next, to investigate how maximum depth of the individual tree affects the runtime of learning, we vary the maximum depth of each individual tree among $\{3, 4, 5, 6, 7, 8\}$ and record the runtime of one boosting stage. As depicted by Figure 5 (a), we can see with the increase of the maximum depth of each individual tree, the runtime increases almost linearly. This indicates that we can train a relatively deep tree with comparatively little time, which is very appealing in practice, especially in the scenario of big data.

Finally, we would like to study the impact of data size on the scalability of our proposed system. We augment the feature sets by feature products. As shown in Figure 5 (b) and Figure 5 (c), we investigate the effects of feature number and sample number, respectively. As depicted by Figure 5 (b) and Figure 5 (c), to study the effect of those two variables, we vary the feature number in the range of $\{50, 500, 1000, 5000\}$ and the sample number in $\{5000, 10000, 30000\}$. We fix the maximum depth of the individual regression trees to 3. We compare the runtime of one boosting stage to investigate how each variant affects the efficiency of the algorithm. From the result, we make similar observations on both Figure 5 (b) and Figure 5 (c). The results imply that sample and feature numbers contribute equally to running time. In addition, we can see that our proposed framework scales well even with relatively big data.

Performance of Completely SecureBoost

To investigate performance of *Completely SecureBoost* in both security and prediction accuracy, specifically, we aim to answer the following two questions: (1) Does the first tree, built upon only features held by active party, learns enough information to mask the actual label by residuals? (2) Does the *Completely SecureBoost* suffers from a great loss of performance compared with *SecureBoost*?

First, we study the performance of *Completely SecureBoost* in security. Following the analysis in Section Analysis of Information Leakage, we evaluate information leakage in terms of leaf purity. As discussed in Theorem 3, we know that when the first tree of *Completely SecureBoost* fits the label information well, the residuals won't reveal much label information. Therefore, to verify the security of *Completely SecureBoost*, we have to illustrate that the first tree of *Completely SecureBoost* indeed masks the actual label well. We conduct experiments on two real-world datasets, Credit 1 and Credit 2. As shown in Table 2, we compare the mean leaf purity of the first tree with the second tree. In particular, the mean leaf purity is the weighted average, which is calculated by $\sum_{i=0}^{k} \frac{n_i}{n} p_i$. Here, k represents number of leaves in total. p_i and n_i are defined as leaf purity and number of instances associated with leaf i. n corresponds to number of instances in total. According to Table 2, the mean leaf purity decreases significantly from the first tree to the second tree on both datasets, which validates the effectiveness of *Completely SecureBoost* in information protection. Moreover, the mean leaf purity of the second tree is just over 0.6 on both datasets, which is good enough to prevent the label information from revealing.

Next, to investigate the performance of *Completely SecureBoost* in prediction accuracy, we compare *Completely SecureBoost* with *SecureBoost* with respect to the the first tree's performance and the overall performance. We conduct experiments on datasets, Credit 1 and Credit 2. Both of them involve the task of binary classification. Thus, we consider the commonly used accuracy, Area under the ROC curve (AUC) and f1-score as the evaluation metric. All these three evaluation metric are the higher the better. The results are presented in Table 3. As can be observed, *Completely Se-*

cureBoost performs equally well compared to *SecureBoost* in almost all cases. We also conduct a pairwise Wilcoxon signed-rank test between *Completely SecureBoost* and *SecureBoost*. The comparison results indicate that *Completely SecureBoost* is as accurate as *SecureBoost*, with a significance level of 0.05. The property of lossless can still be guaranteed for *Completely SecureBoost*.

Conclusion

In this paper, we proposed a novel lossless privacy-preserving algorithm, SecureBoost, to train a high-quality tree boosting model when the training data remains secret over multiple parties. We theoretically prove that our proposed framework is as accurate as non-federated gradient tree boosting algorithms that bring all the data into one place naively. Along with a proof of security, we discuss what would be required to make the protocols completely secure. The experimental results show that our proposed SecureBoost scales well even with relatively big data.

We believe that the research in federated learning is just beginning. While in this paper we showed how to adapt a Boosted Tree algorithm to federated learning settings, much remains to be done on other machine-learning algorithms in privacy-preserving and lossless manners. Other encryption algorithms can be considered as well that ensures the above properties.

References

[Abadi et al. 2016] Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318. ACM.

[Albrecht 2016] Albrecht, J. P. 2016. How the gdpr will change the world. *Eur. Data Prot. L. Rev.* 2:287.

[Chen and Guestrin 2016] Chen, T., and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794. ACM.

[Djatmiko et al. 2017] Djatmiko, M.; Hardy, S.; Henecka, W.; Ivey-Law, H.; Ott, M.; Patrini, G.; Smith, G.; Thorne, B.; and Wu, D. 2017. Privacy-preserving entity resolution and logistic regression on encrypted data. *Private and Secure Machine Learning (PSML)*.

[Dwork, Roth, and others 2014] Dwork, C.; Roth, A.; et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9(3–4):211–407.

[Dwork 2008] Dwork, C. 2008. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, 1–19. Springer.

[Friedman et al. 2000] Friedman, J.; Hastie, T.; Tibshirani, R.; 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):337–407.

[Gilad-Bachrach et al. 2016] Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; and Wernsing, J. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, 201–210.

[Goodman and Flaxman 2016] Goodman, B., and Flaxman, S. 2016. European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*.

[Hardy et al. 2017] Hardy, S.; Henecka, W.; Ivey-Law, H.; Nock, R.; Patrini, G.; Smith, G.; and Thorne, B. 2017. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*.

[He et al. 2014] He, X.; Pan, J.; Jin, O.; Xu, T.; Liu, B.; Xu, T.; Shi, Y.; Atallah, A.; Herbrich, R.; Bowers, S.; et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, 1–9. ACM.

[Konečnỳ et al. 2016] Konečnỳ, J.; McMahan, H. B.; Yu, F. X.; Richtárik, P.; Suresh, A. T.; and Bacon, D. 2016. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.

[Li et al. 2017] Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R. P.; Tang, J.; and Liu, H. 2017. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)* 50(6):94.

[Liang and Chawathe 2004] Liang, G., and Chawathe, S. S. 2004. Privacy-preserving inter-database operations. In *International Conference on Intelligence and Security Informatics*, 66–82. Springer.

[Mayer-Schonberger and Padova 2015] Mayer-Schonberger, V., and Padova, Y. 2015. Regime change: Enabling big data through europe's new data protection regulation. *Colum. Sci. & Tech. L. Rev.* 17:315.

[Mohassel and Zhang 2017] Mohassel, P., and Zhang, Y. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *2017 38th IEEE Symposium on Security and Privacy (SP)*, 19–38. IEEE.

[Oentaryo et al. 2014] Oentaryo, R. J.; Lim, E.-P.; Finegold, M.; Lo, D.; Zhu, F.; Phua, C.; Cheu, E.-Y.; Yap, G.-E.; Sim, K.; Nguyen, M. N.; et al. 2014. Detecting click fraud in online advertising: a data mining approach. *Journal of Machine Learning Research* 15(1):99–140.

[Paillier 1999] Paillier, P. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, 223–238. Springer.

[Regulation 2016] Regulation, P. 2016. The general data protection regulation. *European Commission. Available at: https://eur-lex. europa. eu/legal-content/EN/TXT*.

[Rouhani, Riazi, and Koushanfar 2017] Rouhani, B. D.; Riazi, M. S.; and Koushanfar, F. 2017. Deepsecure: Scalable provably-secure deep learning. *arXiv preprint arXiv:1705.08963*.

[Shokri and Shmatikov 2015] Shokri, R., and Shmatikov, V. 2015. Privacy-preserving deep learning. In *Proceedings of*

the 22nd ACM SIGSAC conference on computer and communications security, 1310–1321. ACM.

[Vaidya and Clifton 2005] Vaidya, J., and Clifton, C. 2005. Privacy-preserving decision trees over vertically partitioned data. In *IFIP Annual Conference on Data and Applications Security and Privacy*, 139–152. Springer.

[Vaidya et al. 2008] Vaidya, J.; Clifton, C.; Kantarcioglu, M.; and Patterson, A. S. 2008. Privacy-preserving decision trees over vertically partitioned data. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 2(3):14.

[Vaidya 2008] Vaidya, J. 2008. A survey of privacy-preserving methods across vertically partitioned data. In *Privacy-preserving data mining*. Springer. 337–358.