Computers, Materials & Continua
DOI: 10.32604/cmc.2023.031684
Article

Federation Boosting Tree for Originator Rights Protection

Yinggang Sun[1], Hongguo Zhang[1], Chao Ma[1,*], Hai Huang[1], Dongyang Zhan[2,3] and Jiaxing Qu[4]

[1]Harbin University of Science and Technology, Harbin, 150040, China
[2]School of Cyberspace Science, Harbin Institute of Technology, Harbin, 150001, China
[3]The Ohio State University, Columbus, 43202, USA
[4]Heilongjiang Province Cyberspace Research Center, Harbin, 150001, China
*Corresponding Author: Chao Ma. Email: machao8396@163.com
Received: 24 April 2022; Accepted: 29 June 2022

Abstract: The problem of data island hinders the application of big data in artificial intelligence model training, so researchers propose a federated learning framework. It enables model training without having to centralize all data in a central storage point. In the current horizontal federated learning scheme, each participant gets the final jointly trained model. No solution is proposed for scenarios where participants only provide training data in exchange for benefits, but do not care about the final jointly trained model. Therefore, this paper proposes a new boosted tree algorithm, called RPBT (the originator Rights Protected federated Boosted Tree algorithm). Compared with the current horizontal federal learning algorithm, each participant will obtain the final jointly trained model. RPBT can guarantee that the local data of the participants will not be leaked, while the final jointly trained model cannot be obtained. It is worth mentioning that, from the perspective of the participants, the scheme uses the batch idea to make the participants participate in the training in random batches. Therefore, this scheme is more suitable for scenarios where a large number of participants are jointly modeling. Furthermore, a small number of participants will not actually participate in the joint training process. Therefore, the proposed scheme is more secure. Theoretical analysis and experimental evaluations show that RPBT is secure, accurate and efficient.

Keywords: Federated learning; data privacy; rights protection; decision tree

1 Introduction

With the increasing awareness of large companies on data security and user privacy protection, users have serious concern on their private information, which may be leaked or even abused by others for commercial or political purposes [1]. Each data owner's private dataset may contain sensitive information that cannot be made public, and direct exposure may violate some privacy policies, such as the General Data Protection Regulation implemented by the EU [2]. Enterprises and regulators have begun to think about how to ensure the legal compliance of data circulation and avoid risks such

as illegal data transfer and resale [3]. Data owners only allow data to be kept in their own hands, which makes the data circulation difficult [4], thus forming data island.

To tackle data island issue in the condition of data, the federated learning framework is proposed by Google in 2016 [5]. Its basic idea is that each participant with the data source trains a model, and uploads the local model instead of the original data to the central server that aggregates the model. The central server obtains the global model through model aggregation, and then distributes the global model to each participant for local model update, and finally completes joint training. This framework is also known as the horizontal federated learning framework [6]. At present, for the horizontal joint learning usage scenario, there is equality among all participants. Each party provides its own data for joint modeling, and finally everyone gets a global model. However, in the horizontal federation process, some participants only want to profit from the training data provided and do not need to obtain the global model. That is, the scenario where one party initiates joint modeling by purchasing data from other participants, but only the initiator ends up with access to the global model has not been solved.

In response to the problems mentioned above, this paper proposes an efficient and privacy-preserving horizontal federated modeling algorithm RPBT, which enables the originator to independently obtain the final joint training model while protecting the data privacy of the participants. Under the premise of ensuring the data privacy of each participant, only the originator can obtain the final global model. The contribution of this paper can be summarized into the following points:

- We presented to ensure the confidentiality and security of participants data and to protect the rights and interests of originator during the joint training process.
- We presented an efficient boosting tree model method RPBT for joint training, which applies the idea of data batch training in the process of machine learning to the batch selection of participants, so that the participants cannot obtain a complete joint model, which ensures that the originator independently acquires the rights of the joint model.
- We perform a very detailed theoretical and experimental analysis of the built RPBT model, and consider our method to be safe, accurate and effective.

The rest of the paper is organized as follows: In Section 2, we give the system model, threat model and problem definition. In Section 3, we briefly introduce necessary preliminaries. In Section 4, we describe in detail the federation boosting tree model of the originator's rights protection. In Section 5, we give the theoretical analysis, including correctness and security. In Section 6, we conduct the actual experimental analysis. In Section 7, we review the relevant literature. Finally, we summarize the paper.

2 Model and Problem Definition

2.1 System Model

Our system model considers a federated boosted tree model of originator rights protection. All participants agreed to share the dataset for training the global model, but do not want to reveal their private data to anyone. At the same time, only the originator can obtain the final joint model, which protects the rights of the originator. As shown in Fig. 1, the model framework consists of three parts:

- *Originator*. The final joint model is obtained by purchasing participant data for joint training.
- *Participant*. Provide a dataset to cooperate with the originator for model training, and the final joint model is unknown.

- *Coordinator*. Using the secure aggregation protocol, under the premise of protecting data privacy, the model information of the originator and participants is aggregated, and the final model information is sent to the originator.

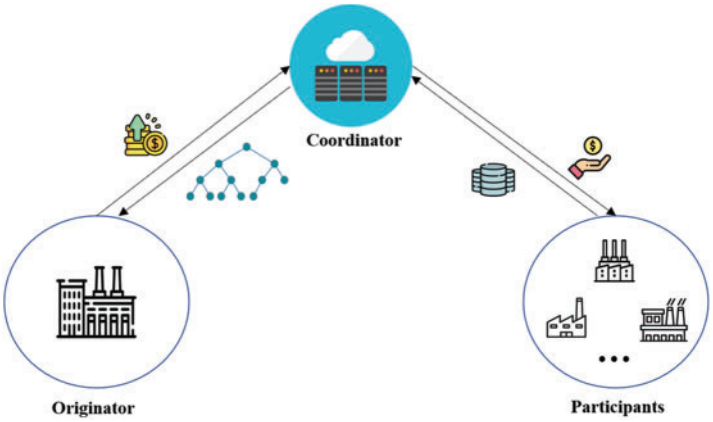


Figure 1: The framework of RPBT

2.2 Threat Model

Assuming that the coordinator is completely credible, no external user can truncate or tamper with the communication between it and the participants. In addition, the coordinator itself will not leak the data information in the process of constructing the lifting tree model to the unknown user.

All participants are semi-honest. He executes the specified protocol honestly, but may launch large-scale reasoning attacks. That is, some participants may use untrue data to participate in model training and disrupt the model effect. It is also possible that a group of participants colluded to infer the data information of other participants.

We also assume that the coordinator will not collude with the participants. Such a non-conspiracy assumption is reasonable in practice, because the coordinator and participants will maintain their reputation, is unlikely to collude with others to undermine reputation.

2.3 Problem Definition and Design Objectives

What we are concerned about is building a boosting tree model for the protection of the rights of the originators on the premise of ensuring the data privacy of all participants. As shown in Fig. 1, it is assumed that both the originator and the participant are factories that produce the same kind of product. If the originator wants to predict the quality of the product from the machine production parameters, then a large amount of production data is needed to build a predictive model. We designed a federal training platform, the originator purchases production data of the participating factories with money. After the participants receive the money, they provide the data to cooperate with the originator for joint training. Let the originator get the final joint model to predict the quality of the product. Our design goals are as follows:

- *Efficiency*: RPBT supports joint training of large-scale participants. Use parallel computing to optimize computing time for large-scale participant model training.
- *Privacy Protection*: To protect the data privacy of all participants, the participants calculate the model information through their own data, and add noise to the model information when

sending it to the collaborating parties. After the coordinator accepts the participant model information, it will use the secure aggregation protocol to aggregate all participant model related information in a secure manner.
– *Right Protection*: As the parties providing data to participate in training, they will not get the final boosted tree model. Only the originator gets the final joint model, which guarantees the rights and interests of the originator.

3 Preliminaries

3.1 Secure Aggregation Protocol

Secure aggregation protocols are a way to use secure multi-party computation (MPC) to securely compute sum of model parameter updates from individual user devices to advance privacy-preserving machine learning [7]. The protocol allows the server to compute the sum of data vectors held from multiple participants in a secure manner (i.e., without learning each user's individual contribution). The core idea is to add an obfuscation term to each client's gradient. Since the obfuscation term is specially constructed by negotiation with other clients and known only to itself, the obfuscation term in each client's gradient passed to the server can be eliminated during aggregation, making it impossible for the server to attack the true gradient value of each client while getting the true gradient aggregation result. Since each client sends an obfuscated gradient and the obfuscated terms in the gradient are known only to the client, the privacy of each client's gradient can be guaranteed without collusion between other clients and the server. For example, in a federated learning environment, model updates are provided to deep neural network aggregation users. It has low runtime and communication overhead, even in scenarios with large datasets and many participating parties. The protocol is secure in an honest but curious active adversary setting, and demonstrates that it remains secure even if an arbitrarily chosen subset of users drop out at any time.

3.2 XGBoost

XGBoost [8] is an open source machine learning project developed by Tianqi Chen and others that efficiently implements the GBDT algorithm with many algorithmic and engineering improvements, and has been widely used and achieved good results in Kaggle competitions and many other machine learning competitions. XGBoost is an optimized distributed gradient boosting library designed to be efficient and portable. It implements machine learning algorithms in the Gradient Boosting framework. XGBoost provides parallel tree boosting (also known as GBDT, GBM) to solve many data science problems quickly and accurately. The same code runs on major distributed environments (Hadoop, SGE, MPI) and can solve problems with over billions of samples. XGBoost leverages out-of-core computing and enables data scientists to process hundreds of millions of samples of data on a single host. XGboost uses CART decision trees as sub-models, and Gradient Tree Boosting to integrate the learning of multiple CART trees to obtain the final model. base model supports not only decision trees but also linear models, and here we use an objective function based on the decision Here we use the objective function based on the decision tree.

4 RPBT Scheme

4.1 Overview

In the problem definition, we mentioned the need to devise a scheme that protects the rights of the originator and allows only the originator to obtain the final joint model. As shown in Fig. 2, RPBT proceeds as follows.

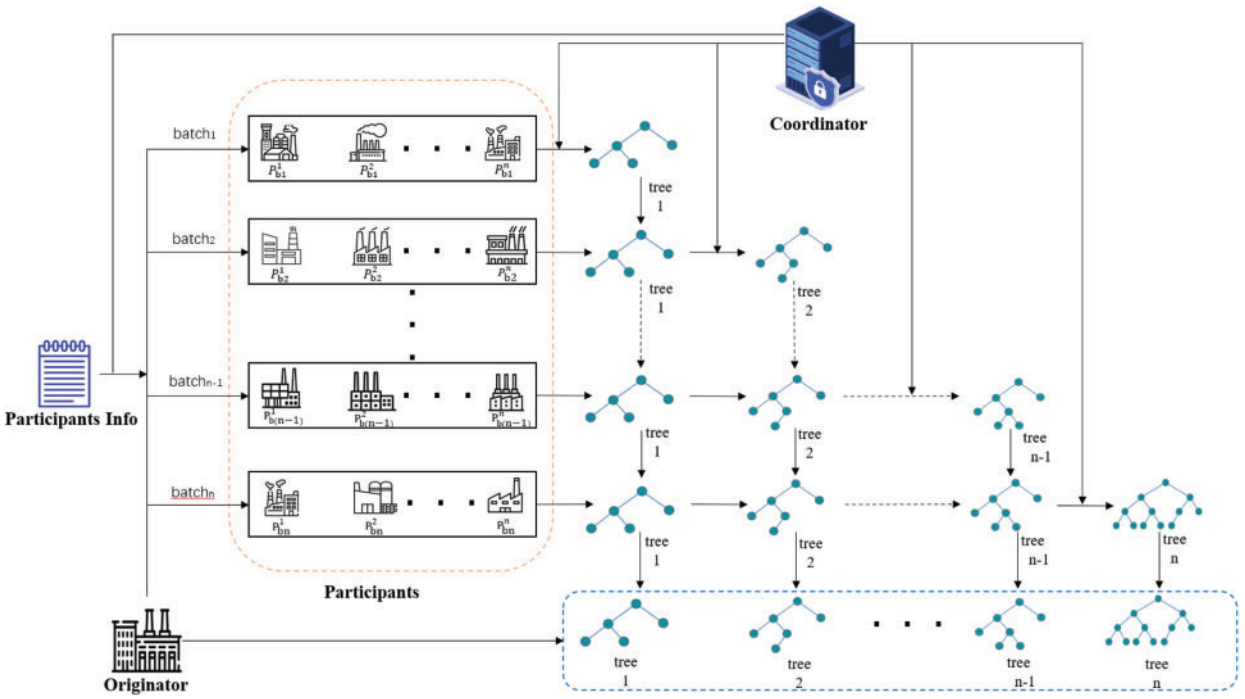


Figure 2: Overview of RPBT

1) **System initialization**. Determine the group information of the participants, set the safety aggregation related parameters and the sponsor rights protection parameters.
2) **Local histogram computation**. Design a local histogram calculation process and ensure data privacy during joint training.
3) **Histogram security aggregation**. The coordinator calculates the optimal splitting point by using the participant histogram information through the secure aggregation method.
4) **Joint model distribution**. After the joint training, the sponsor obtains the joint model independently to ensure the rights and interests of the sponsor.

4.2 System Initialization

Before the task is executed, the information of the participants is counted, and the groups are grouped according to the number of participants and the size of the available data set, so as to facilitate the selection of training batches during joint training. At the same time, the proportion of participants who do not participate in joint training is set according to the number of participants in each group. By giving a fake intermediate model to the participants who do not participate in the joint training, the participants are not sure whether the intermediate model obtained by themselves is real, and the participants cannot use the intermediate model.

4.3 Local Histograms Computation

The calculation process of joint modeling is designed through the basic idea of horizontal federated learning combined with XGBoost algorithm. Let's first review the specific calculation details of the XGBoost algorithm in the modeling process. A dataset $X \in R^{n \times m}$ with n samples and m features, suppose we have trained K trees and the final predicted value for the i-th sample is equal to:

$$\widehat{y_i} = \phi\left(x_i\right) = \sum_{k=1}^{K} f_k\left(x_i\right) \tag{1}$$

where x_i represents the sample characteristics, $f_k(x_i)$ denotes the prediction result of the *k-th* tree for sample x_i, and finally these values are added together to get the final result $\widehat{y_i}$.

According to the additive model of formula (1), the *t-th* tree is learned based on the *t-th* tree optimization, the residuals are calculated based on the predicted values of the t-1 tree, and then the t residual tree is fitted, and the objective The Taylor second-order expansion of the objective function is as follows:

$$L^{(t)} \simeq \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}^{(t-1)}\right) + g_i f_t\left(x_i\right) + \frac{1}{2} h_i f_t^2\left(x_i\right)\right] + \Omega\left(f_t\right) \tag{2}$$

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

In constructing the *t-th* tree, starting from the root node, for each split, the original one node will split into two nodes, and the sample data in the original node will enter into each of the two nodes according to the judgment rule. After each new split, we need to check whether this split will bring gain to the loss function. When splitting a node, we use the following formula to calculate the gain of the split point, where The IL and IR represent the instance space of the left and right tree nodes after splitting. After obtaining an optimal tree structure, the leaves w_j^* optimal weights can be obtained by the following formula, where I_j is the instance space of leaf j.

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

After obtaining an optimal tree structure, the leaves $w^*{}_j$ optimal weights can be obtained by the following formula, where I_j is the instance space of leaf j.

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

From the above review of XGBoost knowledge, we can observe that computing the split point only depends on g_i and h_i. The calculation steps were designed as follows:

1) The originator and the participants calculate the quantile interval information according to the data feature set they own, and send the quantile interval information to the coordinator.
2) The coordinator integrates all quantile interval information, calculates the average quantile sketch, and distributes it to the originators and participants.
3) The originator and participants calculate the local histogram (including the information of g_i and h_i), and then send the local histogram to the collaborating party.
4) The collaborating party merges the local histograms of the originator and the participating parties into a global histogram. The splitting point gain is calculated according to the global histogram, and the splitting feature and splitting threshold corresponding to the optimal splitting point gain are sent to the originator and the participants.
5) The originator and participants update the local histogram after receiving the splitting feature and splitting threshold.

In our lifting tree algorithm, the originator and the participants send the g_i and h_i information to the coordinator, and the coordinator calculates the gain to determine the optimal split point and the corresponding split threshold. Therefore, during joint training, the originator and the participants

will not expose the original data information. However, if the coordinator directly uses g_i and h_i to calculate L_{split}, it is equivalent to expose the g_i and h_i data information owned by the local histogram of each participant. In order to further ensure the data privacy and security of the participants, we introduce a secure aggregation protocol [9]. We need to ensure that the local histogram information of the participants is kept secret from the coordinator, so before the participant sends the local histogram to the coordinator, we consider encrypting the local histogram information of each party by means of secure aggregation, so as to ensure the coordinator. The local histogram information of each participant is not known specifically.

The details of the joint model computation combined with the secure aggregation algorithm is described as Algorithm 1 in Tab. 1. When tree calculates the gain of splitting features, XGBoost uses a pre-sorted algorithm to handle node splitting. The split point calculated in this way is more accurate, but it also causes a lot of time overhead. To solve this problem, Lightgbm chose a decision tree algorithm based on histogram. Compared with the pre-sorted algorithm, histogram has many advantages in memory consumption and computational cost. We also added the histogram algorithm to the calculation process to speed up the calculation process. The basic principle is to directly pass in a number of buckets, and then use the improved GK-summary algorithm to calculate the quantile ϕ. We calculate the corresponding value in the data based on this quantile. Forms an interval that divides all samples in the matching interval into a bucket. Since the data splitting only relies on the values of G and H, we only need to count the G and H values of each sample to meet the requirements of subsequent tasks.

Table 1: Hist algorithm

Algorithm 1 **Create a historgram with encrypted** $<g_i>, <h_i>_{\{i \in I\}}$

Input: I: training data of one client;
d: feature dimension;
Role list: contains each role name, $< g_i >, < h_i >_{\{i \in I\}}$ calculated by I;
privatekey: generated by the Diffie-Hellman algorithm for noise generation

Output: A historgram with encrypted $< g_i >, < h_i >_{\{i \in I\}}$

begin

1: $H \leftarrow newhistorgram$;
2: **for** $k = 1 \rightarrow m$ **do**;
3: Build bin to H;
4: Calculated by precentile rate $\text{quantile}_1, \ldots\ldots, \text{quantile}_n$ which from GK-summary Algorithm;
5: **end for**
6: **for** $i = 1 \rightarrow d$ **do**
7: **for** $j = 0 \rightarrow bin$ **do**
8: **if** I *in bin range* **then**
9: $H[bin].g \leftarrow I.f[i].g$
10: $H[bin].h \leftarrow I.f[i].h$
11: $H[bin].n \leftarrow H[bin].n + 1$
12: **end if**
13: **end for**

(Continued)

Table 1: Continued

```
14: end for
15: for role uuid, private key in DH_list do
16:         if role uuid > self . roleuuid then
17:             ret = random(private key)
18:             H = H + ret
19:         else
20:             ret = random(private key)
21:             H = H − ret
22:         end if
23: end for
24: return A historgram with encrypted by other's private key
```

Next, we design how to complete data encryption when calculating the local histogram. We choose to use Diffie-Hellman algorithm [10] to generate a private key for the initiator and all participants to generate noise when encrypting. In addition, we set uuid to each participant to show identity, and then judge the way in which the encryption process uses noise according to *uuid*.

4.4 Histogram Security Aggregation

We design the operation of model aggregation to be done on the coordinator side. The specific details are as described in Algorithm 2 in Tab. 2. The most important step for the coordinator is to find the best split point. Our histogram format is $[g, h, n]$, where g is the sum of the first derivatives of all samples in this bucket, and h is two sum of derivatives. Then according to the segmentation scheme of XGBoost, we traverse each bucket of each feature to find the optimal segmentation feature and bucket number. Then it is returned to each participant, who divides locally according to the number of features and buckets. Then calculates a new histogram and repeats the above process to complete the training. During the coordinator's process of finding the bestsplit point, the histogram data of all participating parties are summed. According to the method we have dealt with before, the addition can eliminate the noise in the encrypted data, so that the coordinator can obtain a complete and unencrypted global data without obtaining the original data of any participant.

Table 2: Split algorithm

Algorithm 2 Aggregated encrypted historgrams and find best split point
Input: I: instance set of current node,
H: encrypted historgrams from each client,
L: role list conclude parvate key
Output: best split feature number and bin number
begin
1: $H_1, H_2, H_3, \ldots\ldots, H_n \leftarrow$ random chocice n clicent historgram and remove from role list;
2: $H \leftarrow H_1 + H_2 + H_3 + \ldots.. + H_n$;
3: $gain \leftarrow 0$;

(Continued)

Table 2: Continued

4: **for** $k = 1 \rightarrow m$ **do**
5: $G_L \leftarrow 0, G_R \leftarrow 0$
6: **for** *bin in H* **do**
7: $G_L \leftarrow G_L + H[bin][g], H_L \leftarrow H_L + H[bin][h]$;
8: $G_L \leftarrow G - G_l, H_L \leftarrow H - H_L$
9: $gain \leftarrow max(gain, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$;
10: **end for**
11: **end for**
12: **return** k_{opt} and bin_{opt} to each role when we obtain the max gain

4.5 Joint Model Distribution

We need to consider how to protect the rights of the originator during the joint training process. The originator purchases the participant's dataset for joint training through pricing criteria. In the existing horizontal federation scheme, both the originator and the participants can obtain the final federation model. For the originator, now that he has paid for the data set of the participating parties, he just wants to obtain the joint model by himself. Because of the participating parties, both the data value and the joint model are obtained. This is unfair to the originator and does not effectively protect the rights and interests of the originator. Therefore, we design a horizontal federated learning method suitable for the current scenario, in order to solve the problem of how to ensure the rights and interests of the originator.

The main idea is that before joint modeling, the originator determines the number of participants. In the joint modeling process, the originator randomly selects a batch of participants for joint training each round, and an intermediate model is obtained after this round of training. Then the originator selects a batch of new participants, and on the basis of the intermediate model, continues joint training to obtain a new intermediate model. The joint training process stops when the initiator and all participants have completed training. The originator finally gets a joint model jointly trained by all participants.

The specific details of protecting the rights and interests of the originator during the training process are described as Algorithm 3 in Tab. 3. The idea that the originator randomly selects a batch of participants each time during the training process refers to the data batch training process in machine learning. Batch training is used because there is more training data. Using batch training makes training faster and can get stable results. We regard the data of the participants selected in each round as a batch of data for each iteration of the machine learning process. Since good stable results can be obtained using batch data, our process of training the participants as batch data is also effective, and the resulting model can also be said to be stable. In addition, during the joint training process, we can set some participants not to participate in the real training. Instead, a fake intermediate model is returned directly to the participants during the training process. In this way, the participants are not sure whether the intermediate models they have obtained are real, and the participants cannot use the intermediate models, which is equivalent to protecting the rights and interests of the originators from another aspect.

Table 3: Choice and send

Algorithm 3 R**andom choice host and send completed model**
Input: *role list:* contains each role name
D: data of each role
M: adequate model training by adequate data
Output: A complete model
begin
1: **while** $role! = self.role$
2: {the other party ->each party } ← private key (Diffie-Hellman. Algorithm)
3: Add self role to other role with ***private key*** to a dictionary called DH_list
3: **end while**
4: **for** batch in batches**:**
5: Random(n) in role list
6: $D = \sum_{role1}^{n} D_n \rightarrow f(x_n)$ # training model
7: **end for**
8: Random(n) in role list → fake role list
9: **for I in role list do:**
10: IF I in fake role list:
10: $w_n f(x_n) + b \rightarrow f'(x_n)$
11: I ← $f'(x_n)$ #send uncompleted model
13: else
14: I ← $f(x_n)$ #send completed model
15: **end for**

In this section, we propose a new boosted tree algorithm called the RPBT. It mainly solves the following three problems. First, the data privacy and security of each participant is guaranteed during the joint modeling process. Secondly, when the coordinator performs the model calculation, it is ensured that the model information of the participants will not be leaked. Finally, it is ensured that only the originator can get the final boosting tree model, and the participants cannot know the accurate boosting tree model, which protects the rights and interests of the originator.

5 Theoretical Analysis

5.1 Correctness Analysis

Theorem 1. RPBT is lossless, i.e., RPBT model M and XGBoost model M′ behave the same, provided that the initialization and hyper parameterization of models M and M′ are the same.

Proof. According to formula 3, g_i and h_i are the only information for computing the optimal splitting point. In the modeling process, g_i and h_i are encrypted by a secure aggregation protocol. For example, the encryption of message a of side A is $< a >= a + r$, and the encryption of message b of side B is $< b > = b - r$. Where r is the generated noise. By the definition of secure aggregation, we have $< a > + < b > = a + b$. The proof is as follows:

$$< a > + < b > = (a + r) + (b - r) = a + b \tag{5}$$

Thus, we have $g_{la} = \sum_{i \in I_{La}} g_i$, $g_{lb} = \sum_{i \in I_{Lb}} g_i$. Using secure aggregation protocol encryption is$< g_{la} >= \sum_{i \in I_{La}} (g_i + r)$, $< g_{lb} >= \sum_{i \in I_{Lb}} (g_i - r)$, then $g_l =< g_{la} + g_{lb} >= \sum_{i \in I_L} g_i$. As long as there is the same initialization, the optimal splitting point of the model computed encrypted using the secure aggregation protocol and the model M' computed by XGBoost plaintext are the same, resulting in the same and M'. This guarantees lossless properties.

5.2 Security Analysis

Theorem 2. RPBT guarantees data privacy protection for all participants and safeguards the rights of the originator.

Proof. Participants calculate model-related information g and h from local data, and use a secure aggregation protocol when passing gradient information to collaborators. During the entire federation modeling process, the data of the participants does not go out locally, and the model information passed to the coordinator is also encrypted. These two processes protect the data privacy of the participants. For the originator, it exists in the whole process of federated model training. Only the originator can get the final joint model, so the rights and interests of the originator are guaranteed.

6 Experimental Analysis

Our experiments are performed on the following two public datasets:

Dataset 1[1]: This is a public credit scoring dataset related to the task of estimating whether a user will pay on time. This dataset is a binary dataset, which contains a total of 30,000 pieces of data and 25 attributes.

Dataset 2[2]: The hospitalization information of some COVID-19 patients in hospitals in a certain area is recorded, which is related to the estimated hospitalization time of each patient under different medical conditions. This dataset is a multi-class dataset and it contains a total of 318,000 instances and 18 attributes.

In our experiments, we use 70% of the dataset for training, and then the rest of the data is used to for testing the model. All experiments are performed on 8GB video memory and Intel Core i7-9750H video memory.

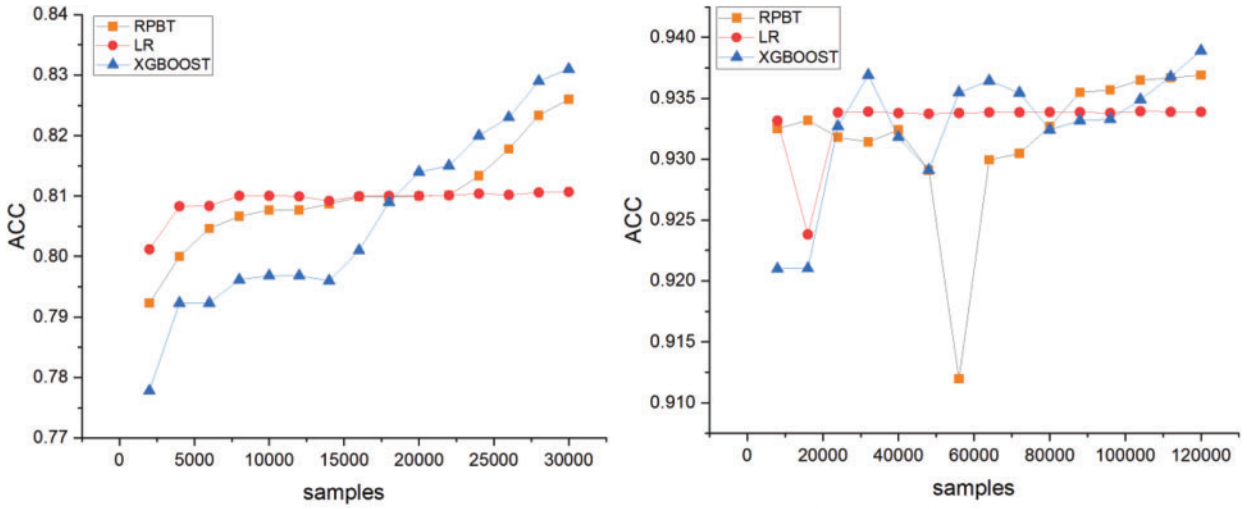
First, we design experiments to compare the performance of RPBT with XGBOOST. When using the RPBT algorithm, we divide the training set into 21 parts in order to simulate the effect of multiple participants getting multiple trees. It is distributed to 1 originator and 20 participants. In each round of training, the originator and two randomly selected participants are trained, and a total of 10 trees are trained. For XGBOOST, we directly use the entire dataset. We set the maximum depth of each tree to 5 and the learning rate to 0.3. We considered commonly used metrics, including accuracy (ACC), F1-Score, and area under the ROC curve (AUC). The results are shown in Tab. 4, and we observe that RPBT performs as well as XGBOOST in almost all cases. There is no big gap between the indicators of RPBT and the indicators of XGBOOST, and some indicators are even better than those of XGBOOST. We can reasonably guess that the RPBT training process did not lose too much model accuracy.

[1] https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset.
[2] https://www.kaggle.com/arashnic/covid19-hospital-treatment.

Table 4: Performance for RPBT *vs*. XGBOOST

	Dataset 1			Dataset 2		
Model	ACC	F1_score	AUC	ACC	F1_score	AUC
RPBT	0.8111	0.3729	0.6115	0.934	0.4485	0.6262
XGBOOST	0.8078	0.3425	0.6006	0.933	0.4528	0.7079

In order to study the accuracy of RPBT in more detail, we added a new logistic regression algorithm for comparison. As shown in Fig. 3, we divided the two datasets into multiple subsets of different sizes. In this case, if the data size of each participant is larger, the final results of binary classification and multi-classification will be more accurate.



(a) binary classification (b) multi-class classification

Figure 3: Accuracy curves for different dataset sizes

In addition, as shown in Figs. 4 and 5, we observe the accuracy of RPBT and XGBOOST by modifying the depth of the tree under the same dataset, and we can still see that RPBT meets our expectations.

In the RPBT scheme, we mentioned that some participants can be set to not conduct joint training.

We can control the number of participants who do not participate in training by modifying the error_host_num parameter specific to the RPBT model. Observe the relationship between error rate and running time during the change of error_host_num. As shown in Fig. 6, when we make the modelmore secure by adding parties that do not train jointly, the error rate of the model increases, but the performance improves.

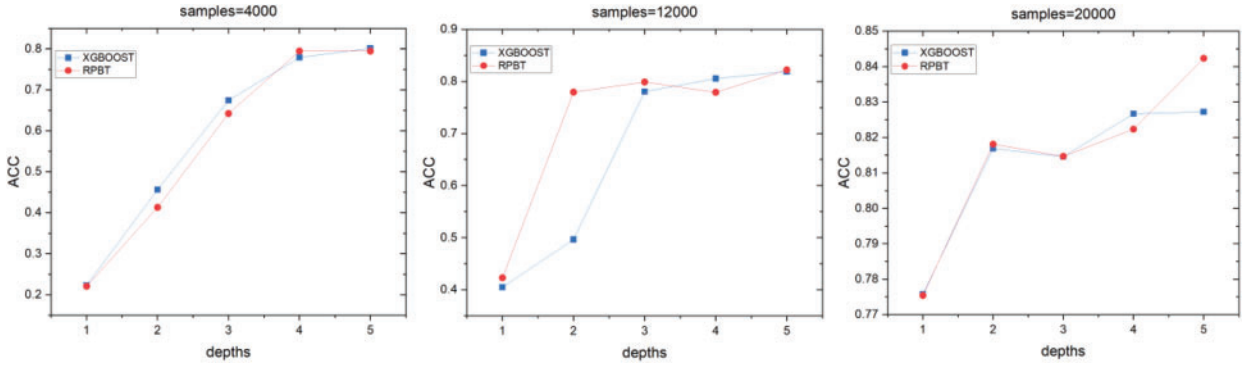


Figure 4: Accuracy at different depths on binary datasets

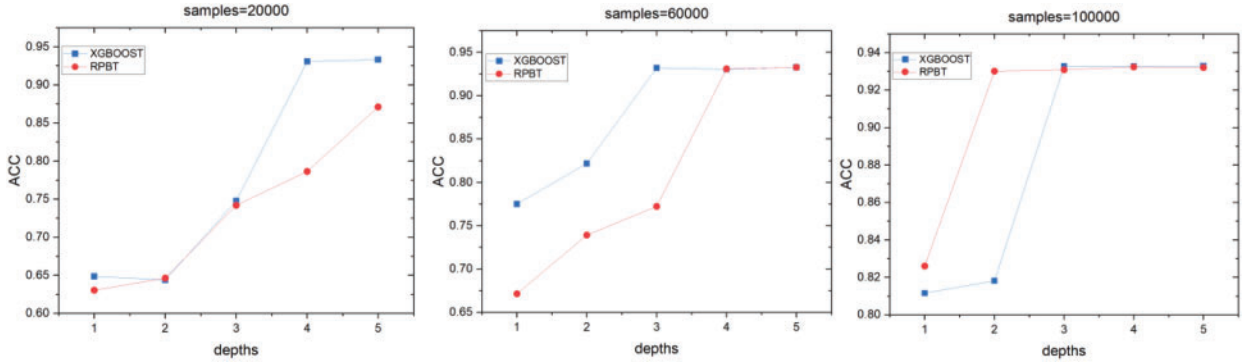


Figure 5: Accuracy at different depths in multi-class datasets

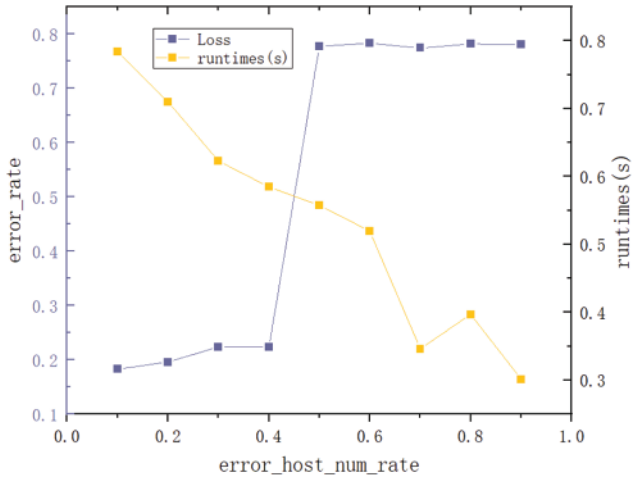


Figure 6: Error rate *vs*. and running time curve under different error_host_num_rate

7 Related Works

The constant occurrence of data leaks and privacy violations has made the public realize the need to protect user privacy and data confidentiality. Recently, researchers are looking at how privacy-preserving technologies can be applied in everyday life. For example, [11,12] proposed several solutions for data privacy protection in the online car-hailing service, and [13,14] proposed a data privacy protection scheme under the Internet of Vehicles on a larger scale. In the context of the dual requirements of data privacy security protection and data silos cracking, federated learning emerges from time to time. To protect the privacy of data when multi-user joint training of models, Google introduced a federated learning framework to build machine learning models based on datasets distributed across multiple devices while preventing data leakage [15]. Its basic idea is that each local mobile terminal uses the local data of the same model architecture to train the local model. The global model can be updated simply by averaging all local models. Following this method proposed by Google, the original machine learning or deep learning model is improved into some new model algorithms to adapt to the federated environment, including decision trees [16], linear [17] and neural networks [18]. Improvements have focused on overcoming statistical challenges and improving the security of federated learning [19], and there are also some research efforts to make federated learning more personalized [20–22]. However, so far, there has not been a scenario where the protection of the rights and interests of the originator has been considered.

The model algorithm we designed is based on the XGBoost algorithm. The exploration of XGBoost can be traced back to the decision tree algorithm, a well-known learning technique for classification and regression tasks. Many decision tree variants have been proposed for different application scenarios, such as C4.5 [23] and CART [24]. Later, the concept of integration was proposed to aggregate multiple trees into one tree, which can be implemented by the gradient boosting tree algorithm [25]. Among them, gradient boosting tree is an efficient and widely used machine learning method. Successful applications in learning to rank [26] and structured prediction [27]. A scalable end-to-end tree boosting algorithm, XGBoost, emerged in later development and is widely used by data scientists to achieve very good model results in many machine learning challenges. However, when the feature dimension is high and the amount of data is large, the efficiency and scalability are still not ideal. One main reason is that, for each feature, they need to scan all data instances to estimate the information gain for all possible split points, which is very time-consuming. In order to solve this problem, we have studied some big data processing and analysis methods according to [28,29]. Meanwhile [30] proposed a new technique that can speed up the training of traditional gradient boosted trees by more than 20 times. We use this computing idea to optimize the computing time of our model.

8 Conclusion

In this paper, we study the issue of initiator rights protection when jointly training models, and propose a federated boosted tree model for initiator rights protection, called RPBT. RPBT can combine multiple participants for joint modeling, while ensuring the rights of the initiator and the data privacy of the participants. In RPBT, we propose an effective method to protect the rights and interests of initiators. When the initiator purchases the data of the participants through payment to build a joint model, we use RPBT so that the participants who provide the sample data will not obtain the final joint model. In addition, the local data of the participants will not be leaked during the joint training process, which ensures the data security of the participants. At the same time, the joint training using the data held by each participant also improves the accuracy of the model. Theoretical analysis

and experiments show that RPBT is secure, accurate and efficient. Future work for the authors or other researchers may be to use the proposed RPBT model in the real world, designing multiple usage scenarios for the RPBT model. And optimize the model during use to improve performance.

Funding Statement: This work was supported by National Natural Science Foundation of China (Grant No. 61976064), the National Natural Science Foundation of China (Grant No. 62172123).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] V. Mayer-Schonberger and Y. Padova, "Regime change: Enabling big data through Europe's new data protection regulation," *Science & Technology Law Review*, vol. 17, no. 2, pp. 315–335, 2015.

[2] R. Zhuo, B. Huffaker and S. Greenstein, "The impact of the general data protection regulation on internet interconnection," *Telecommunications Policy*, vol. 45, no. 2, pp. 102083, 2021.

[3] H. Yu, X. Jia, H. Zhang and J. Shu, "Efficient and privacy-preserving ride matching using exact road distance in online ride hailing services," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 1841–1854, 2020.

[4] H. Yu, H. Zhang, X. Yu, X. Du and M. Guizani, "PGRide: Privacy-preserving group ridesharing matching in online ride hailing services," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5722–5735, 2021.

[5] J. Konecny, H. B. Mcmahan, F. X. Yu, P. Richtarik, A. T. Suresh *et al.,* "Federated learning: Strategies for improving communication efficiency," in *International Conference on Learning Representations*, VAN, CAN, 2016.

[6] Q. Yang, Y. Liu, T. Chen and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.

[7] I. Damgård, D. Escudero, T. Frederiksen, M. Keller, P. Scholl *et al.,* "New primitives for actively-secure MPC over rings with applications to private machine learning," in *IEEE Symp. on Security and Privacy*, Oakland, USA, pp. 1102–1120, 2019.

[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016.

[9] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan *et al.,* "Practical secure aggregation for privacy-preserving machine learning," in *Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security*, Dallas Texas, USA, pp. 1175–1191, 2017.

[10] W. Diffie and M. E. Hellman, "New direction in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 2019.

[11] H. Yu, X. Jia, H. Zhang, X. Yu and J. Shu, "PSRide: Privacy-preserving shared ride matching for online ride hailing systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, pp. 1425–1440, 2021.

[12] H. Yu, J. Shu, X. Jia, H. Zhang and X. Yu, "LpRide: Lightweight and privacy-preserving ride matching over road networks in online ride hailing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 10418–10428, 2019.

[13] Y. Wang, Z. Tian, Y. Sun, X. Du and N. Guizani, "LocJury: An IBN-based location privacy preserving scheme for IoCV," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5028–5037, 2021.

[14] X. R. Zhang, X. Chen, W. Sun and X. Z. He, "Vehicle re-identification model based on optimized densenet121 with joint loss," *Computers, Materials & Continua*, vol. 67, no. 3, pp. 3933–3948, 2021.

[15] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis *et al.,* "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[16] Q. B. Li, Z. Y. Wen and B. S. He, "Practical federated gradient boosting decision trees," *Association for the Advance of Artificial Intelligence*, vol. 34, no. 4, pp. 4642–4649, 2020.
[17] F. Hanzely, S. Hanzely, S. Horváth and P. Richtarik, "Lower bounds and optimal algorithms for personalized federated learning," *Neural Information Processing Systems*, vol. 33, pp. 2304–2315, 2020.
[18] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang *et al.,* "Bayesian nonparametric federated learning of neural networks," in *Int. Conf. on Machine Learning*, LA, USA, pp. 7252–7261, 2019.
[19] V. Smith, C. K. Chiang, M. Sanjabi and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, pp. 4424–4434, 2017.
[20] F. Chen, M. Luo, Z. Dong, Z. Li and X. He, "Federated meta-learning with fast convergence and efficient communication," arXiv preprint, 2018.
[21] Z. Sun, J. Feng, L. Yin, Z. Zhang, R. Li *et al.,* "Fed-dfe: A decentralized function encryption-based privacy-preserving scheme for federated learning," *Computers, Materials & Continua*, vol. 71, no. 1, pp. 1867–1886, 2022.
[22] Z. Gu, L. He, P. Li, P. Sun, J. Shi *et al.,* "Frepd: A robust federated learning framework on variational autoencoder," *Computer Systems Science and Engineering*, vol. 39, no. 3, pp. 307–320, 2021.
[23] S. Sharma, J. Agrawal and S. Sharma, "Classification through machine learning technique: C4.5 algorithm based on various entropies," *International Journal of Computer Applications*, vol. 82, no. 16, pp. 28–32, 2013.
[24] D. Steinberg and P. Colla, "CART: Classification and regression trees," in *The Top Ten Algorithms in Data Mining*, 1st edition, New York, USA: Chapman and Hall/CRC, pp. 193–216, 2009.
[25] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
[26] J. C. Christopher, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, no. 81, pp. 23–581, 2010.
[27] T. Chen, S. Singh, B. Taskar and C. Guestrin, "Efficient second-order gradient boosting for conditional random fields," in *Proc. of 18th Artificial Intelligence and Statistics Conf.*, San Diego, California, USA, vol.1, 2015.
[28] M. A. R. Khan and M. K. Jain, "Feature point detection for repacked android apps," *Intelligent Automation & Soft Computing*, vol. 26, no. 6, pp. 1359–1373, 2020.
[29] N. Binti, M. Ahmad, Z. Mahmoud and R. M. Mehmood, "A pursuit of sustainable privacy protection in big data environment by an optimized clustered-purpose based algorithm," *Intelligent Automation & Soft Computing*, vol. 26, no. 6, pp. 1217–1231, 2020.
[30] G. L. Ke, Q. Meng, T. Finley, T. F. Wang, W. Chen *et al.,* "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, vol. 30, pp. 3146–3154, 2017.