Federated Forest

Yang Liu, Yingting Liu, Zhijie Liu, Yuxuan Liang, Chuishi Meng,
Junbo Zhang, *Member, IEEE*, Yu Zheng, *Senior Member, IEEE*

Abstract—Most real-world data are scattered across different companies or government organizations, and cannot be easily integrated under data privacy and related regulations such as the European Union's General Data Protection Regulation (GDPR) and China' Cyber Security Law. Such *data islands* situation and *data privacy & security* are two major challenges for applications of artificial intelligence. In this paper, we tackle these challenges and propose a privacy-preserving machine learning model, called *Federated Forest*, which is a lossless learning model of the traditional random forest method, i.e., achieving the same level of accuracy as the non-privacy-preserving approach. Based on it, we developed a secure cross-regional machine learning system that allows a learning process to be jointly trained over different regions' clients with the same user samples but different attribute sets, processing the data stored in each of them without exchanging their raw data. A novel prediction algorithm was also proposed which could largely reduce the communication overhead. Experiments on both real-world and UCI data sets demonstrate the performance of the Federated Forest is as accurate as of the non-federated version. The efficiency and robustness of our proposed system had been verified. Overall, our model is practical, scalable and extensible for real-life tasks.

Index Terms—Machine learning, Data mining

1 Introduction

Artificial intelligence has made great progress in recent years thanks to the large amount of data collected in different domains. Unfortunately, the data has also arisen to be the largest bottleneck for the implementation of AI methods. In real-world applications, the big data are scattered across different companies or government organizations and stored in the form of *data islands*, in other words, data across different domains cannot be shared with each other. For companies, the data is among one of the most important assets of companies which cannot be easily shared. Governments' data are highly secured and mostly not utilized. Besides, people now are very sensitive about data privacy. Data breaches happen occasionally and most countries now either have data privacy-related legislation enacted or being drafted. In 2018, the European Union enacted the General Data Protection Regulation (GDPR) [1]. The GDPR provides individuals with more control over their personal data and states strict principles and absolute transparencies on how businesses should handle these data. Any type of tracking or record of personal data must be authorized by the customer before collection and business must clearly state their intentions and plans for the data. For example, profiling is an important application of machine learning and now almost every business uses it for analyzing customers and targeted advertising. The technique itself is neutral and GDPR does not prohibit it. However, the usage of profiling now often causes discrimination on customers, which will not be allowed under GDPR.

Faced with the difficulties and restrictions, the question becomes if it is worth investing in the effort to make use of the scattered data. The answer is yes. Academia, companies and governments could all benefit from resolving the data islands situation. The joint-models are able to improve many current services and products, and support more potential applications, including but not limited to medical study, targeted marketing, urban anomalies detection and risk management, as shown in Figure 1. For example, banks could train joint-models with e-commerce companies to achieve precise customer profiling and improve their marketing strategies. Government organizations could work with ride-hailing companies to have a better understanding of the city's daily traffic flow and adjust the road planing to optimize the traffic during peak hours.

Consequently, the question becomes how can we train the joint-models across different domains or organizations securely. Faced with the challenges of *data islands* and *data privacy & security*, the currently available methods cannot completely solve the problems. Because of this, developing new methods to bridge the gap between real-world applications and data islands becomes an urgent problem. In 2016, a new approach named federated learning [2], [3], [4] was proposed, which mainly focuses on building privacy-preserving machine learning models when data are distributed in different places and cannot be directly collected and stored in one place. A typical application of their work is the word typing prediction on mobile devices. Since the typed words are all private information of the customer, any direct collection is at the risk of violation of laws and regulations, including GDPR. With the federated learning methods, parts of the modeling process can be done on mobile devices and only necessary trained model parameters are uploaded and downloaded to the central

- *Yang Liu, Chuishi Meng, Junbo Zhang and Yu Zheng are with JD Intelligent Cities Business Unit, JD Digits, Beijing, China and JD Intelligent Cities Research, China. Junbo Zhang and Yu Zheng are also affiliated with Institute of Artificial Intelligence, Southwest Jiaotong University, China. Yu Zheng is also affiliated with Xidian University, China. E-mail: {liuyang21cn, msjunbozhang, msyuzheng}@outlook.com, chuishimeng@gmail.com*
- *Yingting Liu is with University of Science and Technology of China. Zhiejie Liu is with Beijing Normal University. Yuxuan Liang is with School of Computing, National University of Singapore. E-mail: yingting6@outlook.com, zhijie_6@163.com, yuxliang@outlook.com*
- *Junbo Zhang is the corresponding author.*

Manuscript received Month Date, Year; revised Month Date, Year.

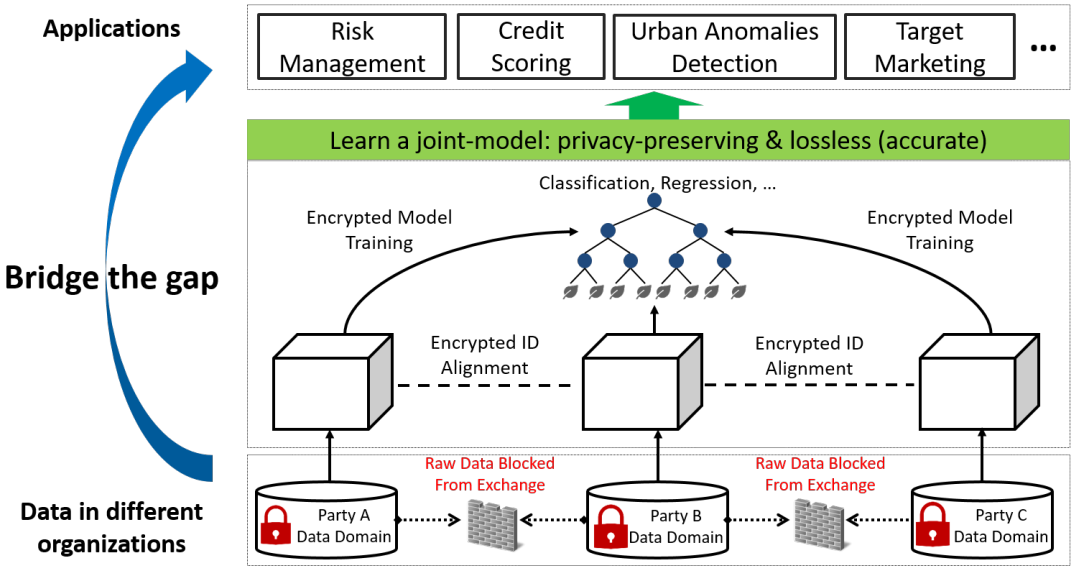


Fig. 1. New Era of Machine Learning

servers, and no privacy was revealed.

Federated learning has provided a new approach to look at the current problems, and shown the possibility of real-life applications. Inspired by their work, we propose a novel privacy-preserving tree-based machine learning model, named *Federated Forest (FF)*. Based on it, we developed a secure cross-regional machine learning system, which is capable of conquering the challenges described above. The core idea here is to distribute the best feature selection process of each tree node on each client, and the master server will collect the local best impurity improvement from all clients and decide which client will provide the best split feature for the current tree node. In this way, a global tree-based model can be built without exchanging any raw data, and each client only has limited and self-related information about this global model, and not knowing anything of other clients. To reduce the communication in prediction, we have taken the advantages of the distributed tree structure and develop a new prediction method. Our contributions are four-folds:

Secured privacy. Data privacy is fully protected by redesigning the tree building algorithms, applying encryption methods and establishing a third-party trusty server. The contents and amount of information exchange are limited to a minimum, and each participant is blind to others.

Lossless (accurate). Our model is based on the methodologies of CART [5] and bagging [6], and fits the vertical federated setting. We experimentally proved that our model can achieve the same level of accuracy as the non-federated approach that brings the data into one place.

Efficiency. An efficient communication mechanism was implemented for the sharing of the intermediate modeling values. A fast prediction algorithm was designed and it's weakly correlated (scale-free) to the number of domains and trees, maximum tree depth and sample size.

Practicability and scalability. Our model supports both classification and regression tasks and is strongly practical, extensible and scalable for real-life applications. The experiments on real-world data sets had proved our model's accuracy, efficiency and robustness.

2 Related Work

In this section, we review the current federated learning and privacy-preserving methods and give the problem formulation.

2.1 Federated Learning

Federated learning [2], [3], [4] was first proposed to solve the problems that rich data are generated from user devices, but due to regulations, it's difficult to build models from the data. The solution is to keep the data on user devices and train a shared model by aggregating locally calculated intermediate results in neural networks. In [7] they proposed a new recommender system that applies federated learning to meta-learning. Federated learning has also been applied to solve multi-task problems in [8] and a loss-based AdaBoost method was developed in [9]. [10] introduced a vertically-aggregated federated learning method. In their work, each data provider possessed unique features, and sample IDs are aligned between them. They jointly learned a logistic regression model to secure data privacy and keep modeling accurately. In addition, a modular benchmarking framework for federated settings was presented in the work of [11]. Although many research products have been coming out, the definition of federated learning was still blurry until the work of [12]. They categorized current federated learning methods into three types, horizontal federated learning, vertical federated learning and federated transfer learning. Following this survey, the same team introduced a new framework known as secure federated transfer learning [13] to build models for the target-domain party by leveraging

rich labels from the source-domain party, as the data sets of the two parties are different in both sample space and feature space. In [14] they reviewed the tree-boosting method and applied it to the vertical federated setting. A lossless framework was proposed and it was able to keep information of each private data provider from being revealed. In [15] they presented a novel reinforcement learning approach that considers the privacy requirement and builds Q-network for each agent with the help of other agents. To make the federated machine learning more practical, they are pushing to build a Federated AI Ecosystem such that the partners can fully exploit their data's value and promote vertical applications. An IEEE standard *Guide For Architectural Framework And Application Of Federated Machine Learning* [16] was also initialized and is being drafted.

2.2 Data Privacy Protection

In federated learning, there are two major encryption methods applied for protecting data privacy and security, which are differential privacy [17] and homomorphic encryption [18]. The idea of differential privacy is to add properly calibrated noise to the algorithm or the data, with examples including [19], [20]. This approach will not affect computational efficiency too much but may weaken model performance. Homomorphic encryption is a method that supports secure multiplication and addition on encrypted data, and once the result is decrypted, it should match the output of operations on the corresponding raw data. The work of [10], [21], [22] all used this approach. If the encryption algorithm satisfies Equation 1, we call this addition homomorphism. If Equation 2 is satisfied, we call it multiplication homomorphism. If both of them are satisfied, then it is fully homomorphism. Homomorphic encryption also supports the operation shown in Equation 3.

$$Encrypt(x) \quad Encrypt(y) = Encrypt(x + y) \tag{1}$$

$$Encrypt(x) \quad Encrypt(y) = Encrypt(x \quad y) \tag{2}$$

$$x \quad Encrypt(y) = Encrypt(x \quad y) \tag{3}$$

There are two major drawbacks of homomorphic encryption. First, the complexity of the algorithm is high and it will be intensely time-consuming for frequent use. Second, it does not support operations of non-polynomial functions very well, such as Sigmoid and Logarithmic function, and approximations are necessary. In the work of [10] they used the Taylor expansion to approximate the Sigmoid function and [23] used the least-squares method. In theory, these approaches could work but in our practice, the results were not ideal.

2.3 Random Forest

Random Forest (RF) is an ensemble supervised machine learning technique, which is widely applied in both classification and regression tasks. In general, RF uses the decision tree as the base classifier and generates multiple decision trees to make predictions [24], where the randomization is presented in two different ways: bagging strategy and random selection of input features. However, building a RF model requires private individuals' data. Such private data is uploaded to a centralized server to extract patterns, and build models from them. To tackle this issue of privacy-preserving, a decision-tree classifier [25] was designed for two parties having their own private database by using the ID3 learning strategy. While this study mainly considered horizontally partitioned data, there were several works that focused on learning tree-based models from vertically partitioned data (i.e., stored in different data sources), such as [26]. Due to the extremely slow speed of existing cryptography-based works for privacy-preserving ML techniques, [27] first showed that RF could be naturally applicable in a fully distributed architecture, and then developed protocols for RF to enable general and efficient distributed privacy-preserving knowledge discovery. Recently, [28] followed the "locally learn then merge" paradigm in cloud computing and extended it to RF models. They proposed ad-hoc procedures for the model encryption (offline) and the decision-tree evaluation (online), which can be seen as a privacy-preserving scoring algorithm for RFs. To the best of our knowledge, in this study, we present the first attempt to combine RF with the concepts of federate learning for protecting the data privacy among different data domains.

3 Problem Formulation

3.1 Data Distribution

In our work, we focus on the vertical federated learning problems, in which all participants have the same sample space but different feature space, as shown in Figure 2. Consider each company or government organization as a regional data domain, denoted as D_i, then the overall data domain is $D = D_1 [D_2 [\quad [D_M$, where $1 \quad i \quad M$. M is the number of regional domains. We denote the feature space of D_i as F_i, then the entire feature space F is $F = F_1 [F_2 [\quad [F_M$. During the modeling process, all features' true names were encoded to protect privacy. For any i and j, if $i \neq j$ and $1 \quad i; j \quad M$, then $F_i \setminus F_j = \varnothing$. In our work, all domains have the same number of samples and the sample IDs were aligned across domains. One master machine was deployed as the parameter server and multiple client machines were used, where each contains one regional data domain. The labels y were provided by one of the clients, which we assume to be the client 1. Then the labels were copied to the master and clients in encrypted forms. Two things to notice here: 1) In reality, M is usually small and even $M = 5$ means there are five different organizations (i.e., government departments, banks, insurance companies, etc.) modeling together, which could be rare. The model design can be totally different for large M. 2) ID alignment is important for tasks such as building loan decision model with personal data from both banks and government, with social security number as ID. However, we are not going to talk about the methods of ID alignment since it is another research topic, discussed in work such as [29]. The notations appeared in this paper are also shown in Table 1.

3.2 Problem Statement

The formal statement of the problem is given as below:

Given: Regional domain D_i and encrypted label y on each client i, $1 \quad i \quad M$.

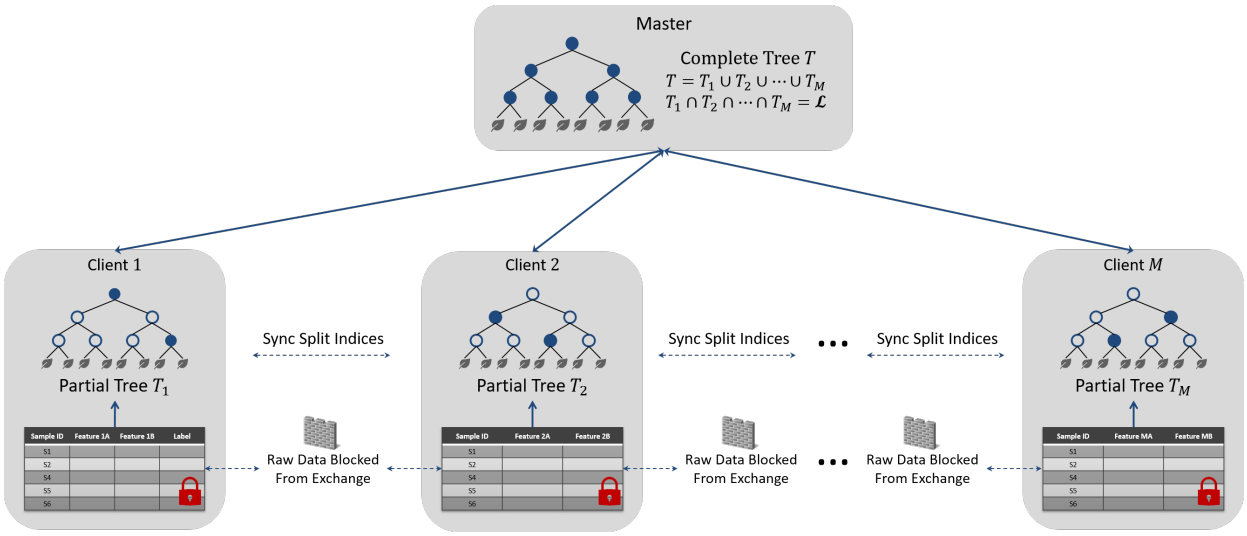


Fig. 2. Federated Forest

Learn: A Federated Forest, such that for each tree in the forest: 1) a complete tree model T is held on master; 2) a partial tree model T_i is stored on each client i, 1 i M.

Constraint: The performance (accuracy, f1-score, MSE, e.t.c.) of the Federated Forest must be comparable to the non-federated random forest.

3.3 Notations

Sample IDs are denoted as S, and S_i^l contains the sample IDs which fall into leaf l of tree T_i. S^l denotes the sample set of leaf node l in the complete binary tree model T.

The test sample set is H, and the single sample is h 2 H.

W_i is the set of decision making paths of sample h that goes through the binary tree to fall into the leaf node of T_i. For the tree T_i, it is possible that h falls into more than one leaf, due to our model storage strategy.

w is the decision making path of the sample h that goes through the complete binary tree to falls into the leaf node in T. For the complete tree T, if sample h fall into one leaf, then it cannot fall into another leaf. It means that any leaf l and g in T, $S^l \setminus S^g = ;$. The complete tree T on master is defined as T = T_1 [T_2 [[T_M.

The detailed descriptions of notations are shown in Table 1.

TABLE 1
Notations

Notation	Description
M	number of regional domains
T_i	partial decision/regression tree stored on ith client
T	complete tree $T = T_1 \cup T_2 \cup \cdots \cup T_M$
T_{i_left}	left subtree of given T_i
T_{i_right}	right subtree of given T_i
$\mathcal{D}_i$	data set held by client i
$\mathcal{D}_{i\_left}$	subset of $\mathcal{D}_i$ that fall into left subtree of given T_i
$\mathcal{D}_{i\_right}$	subset of $\mathcal{D}_i$ that fall into right subtree of given T_i
N	total number of samples in training
$\mathcal{D}$	entire data set $\mathcal{D} = \{\mathcal{D}_1; \mathcal{D}_2; \cdots; \mathcal{D}_M\}$
$\mathcal{F}_i$	feature space of $\mathcal{D}_i$
$\mathcal{F}$	entire feature space of $\mathcal{D}$, $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_M$
y	labels
$\mathcal{L}$	leaf nodes set of the entire tree
$l; g$	leaf node of the current tree, $l; g \in \mathcal{L}$
O	lowest common ancestor of $l; g$ in T
S	the sample IDs of entire data set $\mathcal{D}$
S_i^l	the sample IDs which fall into leaf l of tree T_i
S^l	the sample IDs which fall into leaf l of complete tree T
h	single test sample
$\mathcal{H}$	entire test sample set
W_i	the set of decision making paths of sample h on T_i
w^*	decision making path of sample h on T
k	maxmium tree depth

4 Methodology

In this section, we first give an overview of the framework. Then we present the Federated Forest model and a novel prediction method. Lastly, we will discuss data privacy security methods and analyze the time and communication complexity.

4.1 Framework Overview

Here we present the framework of Federated Forest, which is based on the CART tree [5] and bagging [6], and is able to deal with both classification and regression problems, as shown in Figure 2.

In order to solve the challenges mentioned in section 1, the random forest method is chosen for its natural advantages. It has been proven effective in many applications and is often used as a baseline. Besides, model interpretability

is important when working with government organizations or financial companies, where black box models are not seen favorably. Here we present the framework of Federated Forest, as illustrated in Figure 2.

The master randomly selects samples and features from the entire distribution, then sends them to each corresponding client. The clients calculate the impurity improvement p for each local feature, then encrypt the local maximum p and send it to the master. The master will decrypt the received local maximum impurity improvements, select the best split feature corresponding to the global maximum impurity improvement, and notify the specific client. The client who has the best split feature will split the samples for the left and right subtrees. Then the split sample IDs are sent to the master for distribution. The above operations are recursively applied until stopped. Each tree T_i only keeps the tree structure but not all details of the tree nodes, except when the split feature is contributed locally. Details of the algorithms are given in the following subsections.

4.2 Model Building

4.2.1 Algorithm.

In our work, each tree is built by all parties working together and the tree structure is stored on the master node and every client. However, each tree only stores the split information with respect to its own features. We first present the client-side Federated Forest algorithm in Algorithm 1, and in Algorithm 2 we described how the master coordinates the modeling process.

Following the bagging paradigm, the master node first randomly selects a subset of features and samples from the entire data. Then the master will notify each client of the selected features and sample IDs privately. For the selected features, the master will notify each client privately. For example, if ten features are chosen by the master and client 1 only possesses three of them, then client 1 will only know these three features were selected. It will never know how many features were chosen globally, not to mention what the features were. During the tree construction, the pre-pruning conditions are frequently checked. If the conditions are satisfied, the clients and master will create leaf nodes accordingly.

If the termination condition is not triggered, all clients enter the splitting state, and the best split feature of the current tree node will be selected by comparing the impurity improvements. First, each client i finds the local optimal split feature f_i. Then the master collects all local optimal features and corresponding impurity improvements, allowing the global best feature to be found. Second, the master notifies the client who provided the global best feature. The corresponding client will split the samples and send the data partition results (sample IDs that fall into left and right subtrees) to the master for distribution. For the current tree node, only the client that provides the best split feature will save the details of this split. The other clients are only aware that the selected feature is not contributed by themselves. The split information such as threshold and split feature are also unknown to them. Last, the subtrees are recursively created and the current tree node is returned. In modeling, if the child trees nodes are created successfully, the parent node doesn't need to save the sample IDs for the subtrees. Otherwise, if the connection is down, the modeling can be easily recovered from the breakpoint.

ALGORITHM 1: Federated Forest – Client

Input : Data set D_i on client i;
Local features F_i = ; or $F_i = \{f_A, f_B, \ldots\}$;
Encrypted label y;
Output: Partial Federated Forest Model on Client i

while *tree_build is True* **do**
  Receive F_i^0 F_i and D_i^0 D_i for current tree building;
  Function `TreeBuild` (D_i^0, F_i^0, y)
    Create empty tree node;
    if *the pre-pruning condition is satisfied* **then**
      Mark current node as leaf node;
      /* For classification problems */
      Assign leaf label by voting;
      /* For regression problems */
      Assign leaf label by averaging;
      return *leaf node*;
    end
    p; f 1 ; None;
    if F_i^0 6= ; **then**
      Compute impurity improvement p for any $f \in F_i^0$ and find local maximum p_i;
      Record local best split feature f and split threshold;
    end
    Send encrypted p_i to master;
    if *receive the split message from master* **then**
      /* Global best split feature is from itself */
      is_selected True;
      Split samples and send sample indices of left and right subtrees to master;
    else
      Receive sample indices of left and right subtrees;
    end
    left_subtree `TreeBuild` $(D_{i\_left}^0, F_i^0, y_{left})$;
    right_subtree `TreeBuild` $(D_{i\_right}^0, F_i^0, y_{right})$;
    if *is_selected is True* **then**
      Save f and split threshold to tree node;
    end
    Save subtrees to tree node;
    return *tree node;*
  end
  Append current tree to forest;
end
return *Partial Federated Forest Model on Client i;*

4.2.2 Model Storage.

A tree predictive model is composed of two parts, tree structure and split information such as feature and threshold used for each split. Since the forest is built with all clients working together, the structure of each tree on every client is the same. However, for a given tree node, the client may

ALGORITHM 2: Federated Forest – Master

Input : Indices of D;
Encoded features $F = F_1 \cup F_2 \cup \cdots \cup F_M$;
Encrypted label y;
Output: Complete Federated Forest Model

/*Build trees for forest recurrently*/
while *tree_build is True* **do**
  Broadcast randomly selected samples D';
  Randomly select features F'_i from F_i and send to client i;
  Function `TreeBuild` (D', F', y)
    Create empty tree node;
    if *the pre-pruning condition is satisfied* **then**
      Mark current node as leaf node;
      /* For classification problems */
      Assign leaf label by voting;
      /* For regression problems */
      Assign leaf label by averaging;
      return *leaf node*;
    end
    Receive encrypted $\{pg_{i=1}^{M}$ and related information from all clients;
    Take $j = argmax(\{pg_{i=1}^{M})$ and notify client j;
    Receive split indices from client j and broadcast;
    left_subtree `TreeBuild` $(D'_{left}; F'; y_{left})$;
    right_subtree `TreeBuild` $(D'_{right}; F'; y_{right})$;
    Save subtrees and split info to tree node;
    return *tree node;*
  end
  Append current tree to forest;
end
return *Complete Federated Forest Model*;

or may not store the detailed information. Only the master server can optionally store the complete model. For each tree node, the client will store the corresponding split threshold only if it provided the split feature. If not, the client will store nothing at the current node but only keep the node structure. We denoted the complete tree nodes as T, the one saved on the master, and denoted the tree nodes without full details stored by ith client as T_i. Since the tree structure is consistent, we consider $T_i \subset T$, and $T_1 \setminus T_2 \setminus \cdots \setminus T_M = L$, where L is the leaf node sets. The complete tree T is the union of all partial trees, that $T = T_1 \cup T_2 \cup \cdots \cup T_M$.

Let's assume there are two clients and one master, and the complete tree model T is shown in Figure 3a. In Figure 3b, the left graph is the visualization of tree model T_1 on client 1. Client 1 owns the selected feature *Age*. Therefore it stores the split feature and threshold on node 2. In the meantime, it knows nothing of node 1 but only stores the structure. Right graph shows the tree model T_2 on client 2. Client 2 knows nothing of node 2, but feature *Profession* is known. Then the details related to *Profession* are saved on Node 1 of T_2.

4.3 Model Prediction

Under the vertical federated setting [12], the classical approach of prediction involves multiple rounds of communication between the master and clients, even for only one sample. When the number of trees, maximum tree depth and sample size are large, the communication requirements for predicting will become a serious burden. To address this problem, we designed a novel prediction method that takes advantage of our distributed model storage strategy. Our method only needs one round of collective communication for each tree and even for the overall forest. We first present the prediction algorithm of the client side in Algorithm 3, and in Algorithm 4, we described how the master server coordinates each client to achieve the final predictions.

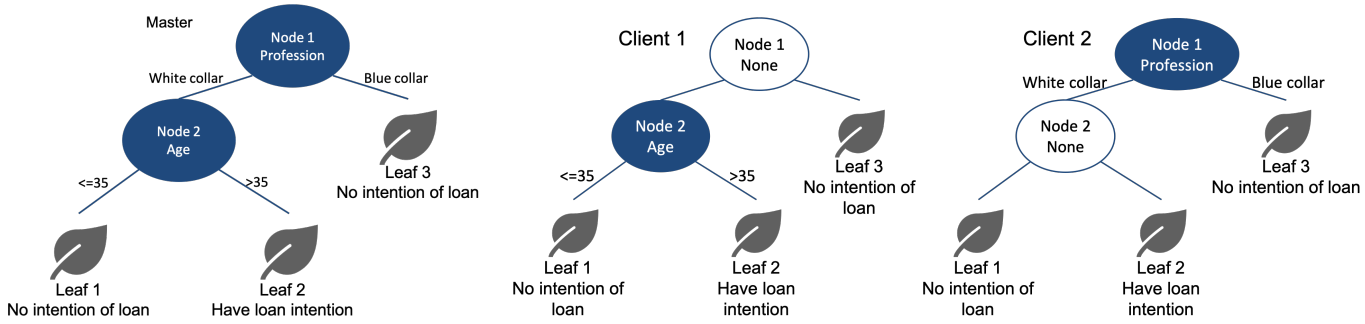
ALGORITHM 3: Federated Forest Prediction – Client

Input : Partial federated forest model saved on ith client;
Encoded features F_i on ith client;
Test set D_i^{test} on ith client;
Output: Samples IDs S_i^l of leaf l on T_i, $l \in L$

while *TreePrediction is True* **do**
  Function `TreePredict` (T_i, D_i^{test}, F_i)
    if *current tree node is leaf node* **then**
      Return sample IDs S_i^l and leaf label;
    else
      if T_i *keeps the split info of current node* **then**
        Split samples into $D_{i\_left}^{test}$ and $D_{i\_right}^{test}$;
        left_subtree `TreePredict` $(T_{i\_left}, F_i, D_{i\_left}^{test})$;
        right_subtree `TreePredict` $(T_{i\_right}, F_i, D_{i\_right}^{test})$;
      else
        left_subtree `TreePredict` $(T_{i\_left}, F_i, D_i^{test})$;
        right_subtree `TreePredict` $(T_{i\_right}, F_i, D_i^{test})$;
      end
      Return left and right subtrees;
    end
    Send $S_i = \{S_i^1; S_i^2; \cdots; S_i^l; \cdots\}$ to master;
  end
end
return;

First, each client uses the locally stored model to predict samples. For the tree T_i on ith client, each sample enters T_i from the root node and finally falls into one or several leaf nodes through the binary tree. When the sample travels through each node, if the model stores the split information at this node, then this sample is determined to enter the left or right subtree by checking the split threshold. If the model does not have split information at this node, the sample simultaneously enters both left and right subtrees.

Again, as shown in Figure 3b, if sample h arrives at node 1 on client 1, it will fall into node 2 and leaf 3 simultaneously because the current node has no split information. For left subtree, sample h arrives at node 2. Assume h['*Age*'] is 40, then it will fall into leaf 2 since bigger than the threshold.



(a) Complete Tree Model on Master (b) Partial Tree Models on Clients

Fig. 3. Tree Models on Master and Clients

ALGORITHM 4: Federated Forest Prediction – Master

Input : Sample IDs S of test set D^{test}
Output: Prediction of Federated Forest
while *TreePrediction is True* **do**
  Gather $\{S_1; S_2; \quad ; S_i; \quad g$;
  Obtain $\{S^1; S^2; \quad ; S^l; \quad g$, where $S^l = S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l$;
  Return label of leaf l for samples in S^l, $l \in L$;
end
/* For classification problems */
Calculate forest predictions by voting on the results of trees;
/* For regression problems */
Calculate forest predictions by averaging the results of trees;
return *Final Predictions*;

Ultimately, sample h will fall into leaf 2 and 3 on client 1. On client 2, assume h['*Profession*'] = 'Blue Collar', then h falls into leaf 3.

Secondly, the path determination of the tree node is performed recursively until each sample falls into one or several leaf nodes. When this process is finished, each leaf node of the tree T_i on client i will keep a batch of samples. We use S_i^l to represent the samples that fall into the leaf node l of the tree model T_i, where $l \in L$. L is the set of leaf nodes of the tree T_i. And the sample IDs of leaf node l in the entire binary tree model T is denoted as S^l.

Thirdly, for each leaf $l \in L$, the master will take the intersection on $\{S_i^l g_{i=1}^M$, and the result will be S^l. Then the sample sets S^l owned by each leaf node on complete tree T are already associated with final predictions. Here we gave a formal proposition on our new prediction method so it can be mathematically defined:

Proposition 1. *For samples S fall into one or multiple leaves on tree T_i, then for any leaf l of the complete tree T, the sample IDs S^l in leaf l can be obtained by taking intersection of $\{S_i^l g_{i=1}^M$, that $S^l = S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l$.*

For the prediction process, samples S will go through the client tree T_i and fall into one or multiple leaves. For any leaf l of the complete tree T, the sample IDs S^l in leaf l can be obtained by taking intersection of $\{S_i^l g_{i=1}^M$, that $S^l = S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l$. The formal proof for this proposition is given at the end of this subsection.

After obtaining the label values for each sample on all trees, we can easily achieve final predictions. In this approach, we only need one round of communication for each tree, or even only one round for the entire forest.

Continue on the example given above, we can find by taking intersection, sample h will finally belong to leaf 3.

Proof. Proof of the Proposition 1.

In order to prove $S^l = S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l$, we will prove:

$$S^l \quad S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l$$
$$S^l \quad S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l$$

Proof of $S^l \quad S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l$:

For any sample h in the leaf l of the complete tree T, $h \in S^l$. w denotes its decision making path from root to leaf node. For model T_i on each client i, if the model stores split information at the current node, it is determined according to the threshold whether this sample enters the left or right subtree. If the current model does not store split information at this node, the sample enters the left and right subtrees simultaneously. Therefore for sample h, its decision making path w on the complete tree T must be a subset of its decision making path W_i on any client i. Then we have $w \quad W_i; 1 \quad i \quad M$, which is equivalent to $h \in S_i^l; 1 \quad i \quad M$. Because of this we can safely say that $h \in S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l$ for any h in S^l. Then we can prove that $S^l \quad S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l$.

Proof of $S^l \quad S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l$:

Assume that sample h doesn't belong to leaf node l but belongs to g in complete model T, which is $h \notin S^l$ and $h \in S^g$. Besides, we assume $h \in S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l$.

$\Rightarrow h \in S_1^g \setminus S_2^g \setminus \quad \setminus S_M^g$, obtained by the above proof.

$\Rightarrow h \in (S_1^g \setminus S_2^g \setminus \quad \setminus S_M^g) \setminus (S_1^l \setminus S_2^l \setminus \quad \setminus S_M^l)$

$\Rightarrow h \in (S_1^g \setminus S_1^l) \setminus (S_2^g \setminus S_2^l) \setminus \quad \setminus (S_M^g \setminus S_M^l)$

That is to say, sample h will fall into the leaf node g and l at the same time in every model stored on the client.

$\because$ In the same binary tree structure, the path from a child node to the root node is fixed and unique.

Under the complete tree structure, the path set of the leaf node g and l up to the root node is $w^l \not\subset w^j$. And the lowest common ancestor node exists and is uniquely set to O.

So $(w^l \not\subset w^j) \quad W_i \Rightarrow (w^l \not\subset w^j) \in (W_1 \setminus W_2 \setminus \quad \setminus W_M)$

So no platform stores the information of the node O.

$\Rightarrow T \not\subset T_1 \not\subset T_2 \not\subset \quad \not\subset T_M$

This contradicts to $T = T_1 \cup T_2 \cup \cdots \cup T_M$.
Therefore the hypothesis doesn't hold.
$\Rightarrow) \ h \in S' \Rightarrow) \ h \in S'_1 \cap S'_2 \cap \cdots \cap S'_M$
$\Rightarrow) \ S' \subseteq S'_1 \cap S'_2 \cap \cdots \cap S'_M$
In summary, we can prove $S' = S'_1 \cap S'_2 \cap \cdots \cap S'_M$. □

4.4 Privacy Protection

In our work, data privacy protection is tightly associated with how the federated forest model is designed.

Here we have categorized our efforts on the privacy protection into five parts:

Identities. In real world tasks, we often face situations where IDs of samples are tied to persons' real identities. Because of this, we have to encrypt the identities before the ID alignment. An example approach could be like the following: First, all clients use an agreed hash method to transform the sample IDs and generate new hashed IDs. Then Message-Digest Algorithm 5 (MD5) can be applied to the hashed IDs and generate irreversibly encrypted IDs.

Labels. For classification problems, even labels are encoded, we could still guess the true values, especially for binary classification. For regression problems, even though labels can be encrypted with homomorphic encryption, it will be extremely time-consuming for modeling. In practical tasks, there will be a trade-off between security protection and computational efficiency.

Features. On each client, local features were encoded before given to the master for global feature sampling. So the master will not know the real meaning of features.

Communication. Encryption methods such as RSA and AES can be applied to secure everything (model intermediate values, sample IDs, e.t.c.) communicated during the training and prediction.

Model Storage. The entire model was distributed across all clients. For each node, the client would store the corresponding split information only if the split feature is on the local machine. If not, it only stored the structure of the current node. Clients knew nothing about each other including whose features were selected, at which tree nodes and the thresholds. Master can optionally keep a copy of the entire model, if the master server is deployed at supervision organizations and also plays the role of auditor.

4.5 Communication Complexity Analysis

Here we give a brief analysis of communication complexity. There are mainly three types of communication during the training, where M is the number of regional domains:

Send and receive. Master sends randomly selected features to each client in every turn for tree building and the client who saves the global optimal feature sends the sample split indices of this feature to master when building the node. The communication complexity is $O(1)$.

Broadcast. Master broadcasts sample indices for each tree node construction. The communication complexity is $O(M)$.

Gather. Master gathers and compares the impurity improvement of features at every turn for node building. It also gathers sample sets of all leaves on each tree stored by clients in the prediction process. The communication complexity is $O(M)$.

Since the maximum depth is k, in a tree, there are at most $2^{k-1} - 1$ intermediate nodes and 2^{k-1} leaf nodes. Take the process of building a tree, for example, the communication complexity of the whole system in the training phase is $O(2^k(M+1))$. For the prediction phase, if not optimized, the communication complexity is $O(2^{k-1}M)$, otherwise, the optimized communication complexity is $O(M)$.

5 Experimental Studies

5.1 Experimental Setup

In this section, we introduce 9 benchmark datasets, including one real-world task (target marketing) and 8 public datasets from UCI [30], [31], [32], [33], as shown in Table 3. The details of the last 8 datasets can be easily accessed at an open website[1], while the *target marketing* dataset was obtained from two data sources to collectively model a user from different views. Among them, one was from an e-commerce company and contains 84 features (e.g., purchasing records and preferences), while the other one was from a bank with 11 user features (e.g., balance and loan information). The target is to discriminate whether a user is a potential customer of a specific service. We have encrypted the sensitive information before collectively modeling.

In the experiments, our model is implemented with Python 3.6, Scikit-learn 0.20, Numpy 1.15.4, python-paillier 1.4.1 and mpi4py 3.0.0. We train/evaluate our model on servers each with 4 CPU cores and Centos 7.0. All the servers are in the same internet environment and the bandwidth is 20m/s. Different sample sizes and feature spaces were considered, and the accuracy, efficiency and robustness of our proposed framework were tested for both classification and regression problems. Notice that we did not pursue absolute accuracy and instead tested whether the performance of our methods is at the same level as the non-federated approach, i.e., lossless. Four main series of experiments were conducted to evaluate our model from different aspects: experiments with two data providers, experiments with multiple data providers, validation of convergence and analysis of prediction efficiency. The details of each test are given in the following subsections.

5.2 Experiments with Two-Party Scenario

In this part, exposed UCI data sets were vertically and randomly separated by feature dimension and placed on two different client servers ($M = 2$), each containing half of the feature space from the original data. For the *target marketing*, it was also placed on two different client servers, of which each contained several business domains. The experiments in this section are summarized as the following:

Federated Logistic/Linear Regression (F-LR): We jointly trained logistic/linear regression models, where data is kept locally and the model is partly stored in each client.

1. https://archive.ics.uci.edu/ml/datasets.php

TABLE 2
Data sets

Classification	Size	Features	Classes
target marketing	156198	95(11/84)	2
ionosphere	351	34	2
spambase	4601	57	2
parkinson [31]	756	754	2
kddcup99	4M	42	23
waveform	5000	21	3
gene	801	20531	5
Regression	Size	Features	Range
year prediction	515345	90	1922-2011
Superconduct [33]	21263	81	0.0002-185

Non-Federated Forest (NonFF): All data were integrated together for Random Forest modeling.
Random Forest 1 (RF1): Partial data from the 1^{st} client was used to build a random forest model.
Random Forest 2 (RF2): Partial data from the 2^{nd} client was used to build a random forest model.
Federated Forest (FF): This is our proposed model, in which two parties jointly learn a random forest. Data were kept locally and model was partly stored in each client.

In the following experiments, we used the grid search to find the best hyperparameters, then we use these parameters to train the model multiple times, and took the average as the final results. All experiments follow the same data sampling strategy to assure consistency, where each data set is divided into a train/test set with an 80/20 ratio and the same random seed is used for different methods. We conducted the experiments on both classification and regression problems, and present the results of accuracy and RMSE in Table 3. We found that the performance of RF1 and RF2 were obviously worse than the NonFF and FF. Both RF1 and RF2 can be considered as modeling with data from one business domain, and the insufficient feature space resulted in an imperfect study of global knowledge. We also found in most tests that the regression models didn't perform very well. For the test on *target marketing*, since direct aggregation of data between two institutions was not allowed, we only ran tests for RF1, RF2, F-LR and FF. The results show that FF performs as expected and better accuracy is achieved by building models on different domains.

For most of the data sets, NonFF and FF outperformed the other methods. In our method, we were building each tree by processing globally on every regional domain, which was the same as the tree built by aggregating raw data together. Z-Test[2] was applied to verify the lossless of our method compared with NonFF, of which the null hypothesis is that the means from two populations are equal at a given level of significance. For each data set, 40 rounds of tests on the NonFF and FF were performed and the *p-value* of each Z-Test is given in Table 3. If the *p-value* 0:05, the null hypothesis cannot be rejected at the 0.05 level and there is no significant difference between the outputs of NonFF and FF. If 0:01 *p-value* < 0:05, the null hypothesis cannot be rejected at the 0.01 level. And statistically, we consider there exists a slight but acceptable difference for this range of *p-value*. The null hypothesis should be rejected if *p-value* < 0:01 with a significant difference between the means. By examining the *p-value* of each data set, we can find that there are six of them proved to have no significant difference between the results of NonFF and FF, and for the rest data sets the differences are slight. No null hypotheses were rejected.

Overall, we can safely confirm that the Federated Forest is a lossless solution for both classification and regression problems, which achieves the same performance as the non-federated random forest.

5.3 Experiments with Multi-Party Scenario

In this part, we ran tests on the *parkinson* data set to verify whether the Federated Forest is capable of conjoining more than two domains effectively and if a reasonable improvement on accuracy could be achieved. We chose *parkinson* to run the test since it already contains eight clearly categorized sub-domains. As for tests of training and prediction efficiency, we duplicated data for ten times. In the tests, each time we added one domain into the federated model, and we recorded the accuracy, training and prediction time. As shown in Figure 4, the accuracy of Federated Forest improved consistently. The training execution time was almost linearly with respect to the number of domains, which is to be expected because all features are to be examined in tree building. For the prediction time, though more domains and features were added, the difference in execution time was negligible. The results demonstrate that our new prediction algorithm is very effective when handling multiple regional domains.

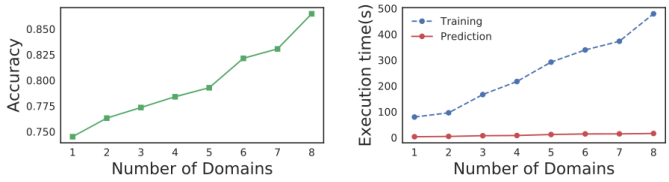


Fig. 4. Accy. & Exec. Time vs. # of Domains

5.4 Convergence Analysis

In this part, we examined the convergence of the federated forest and also compared it with the regular random forest, with the waveform dataset and spambase dataset. For each test, we have checked the accuracy by modeling from one tree up to one hundred trees. The maximum tree depth is set to 6. The results are shown in Figure 5, where the solid lines represent the results of federated forest and the dash liens are results of conventional random forest. As observed, the federated forest can converge by increasing the number of trees. Although it varies from the convergence pattern of random forest, it still reaches a stable condition after a reasonable number of trees. The reason why the federated forest needs more trees to converge is that, in the federated settings, each round of bagging on samples and features

2. Hypothesis Testing: https://online.stat.psu.edu/stat414/node/290/

TABLE 3
Classification experiments

Data set	RF1	RF2	F-LR	NonFF	FF	p-value
target marketing	0.870	0.848	0.862	-	0:890 ± 0:014	-
ionosphere	0.864	0.828	0.873	0:908 ± 0:019	0:896 ± 0:030	**0.211**
spambase	0.844	0.831	0.873	0:943 ± 0:005	0:928 ± 0:020	**0.065**
parkinson [31]	0.849	0.849	0.829	0:859 ± 0:018	0:857 ± 0:013	**0.744**
kdd cup 99	0.974	0.965	-	0:995 ± 0:001	0:995 ± 0:009	0:012
waveform	0.745	0.743	-	0:826 ± 0:008	0:822 ± 0:012	0.029
gene	0.975	0.975	-	0:988 ± 0:005	0:982 ± 0:006	**0.229**

TABLE 4
Regression experiments

Data set	RF1	RF2	F-LR	NonFF	FF	p-value
year prediction	10.47	10.72	9.56	9:537 ± 0:003	9:555 ± 0:061	**0.058**
Superconduct [33]	19.74	17.49	17.52	15:369 ± 0:118	15:411 ± 0:163	**0.186**

can have more impact on the performance than in a non-federated approach. Overall, the convergence is validated.

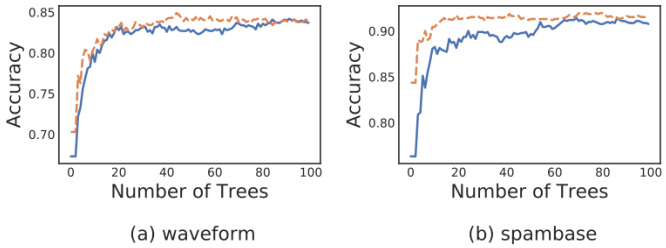


Fig. 5. Convergence

5.5 Prediction Efficiency

In this part, we compared the efficiency of our new prediction method with the classical prediction approach. We used *target marketing*, *spambase* and *waveform* data sets as the examples. We ran all the tests for 20 times and report the average results, as shown in Figures 6, 7 and 8. The solid lines with the dot marker represent the results of the classical prediction method, and the dash lines with x marker represent our proposed prediction method.

Firstly, we set the maximum tree depth to 4 and changed the number of estimators from 8 to 32, and the results were shown in Figure 6. It can be seen that our method produced a strong improvement in prediction efficiency. Though the execution time of both methods increased linearly respect to the number of estimators, the slope varied dramatically between our method and the classical prediction method. For the classical method, there are multiple rounds of communication in each node during prediction. But in our method, there is only one round of communication for each tree.

Secondly, we set the number of estimators to 8, and adjusted the maximum tree depth from 4 to 16. As shown in Figure 7, our method outperformed the classical prediction method again. By increasing the maximum tree depth, the

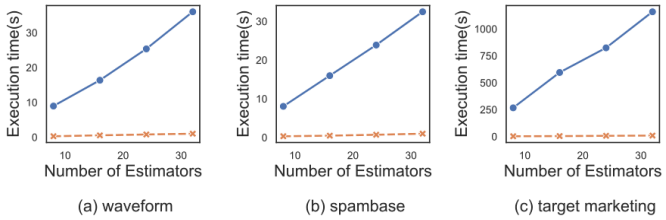


Fig. 6. Prediction Time vs. Number of Estimators

growth rate of prediction time for both methods gradually slowed down and stabilized. This is because by setting the maximum depth to a large number, the tree building may early stop due to pre-pruning and the actual tree depth will be smaller. In our method, no matter how deep the tree is or how many leaf nodes are created, communication was only executed once for each tree.

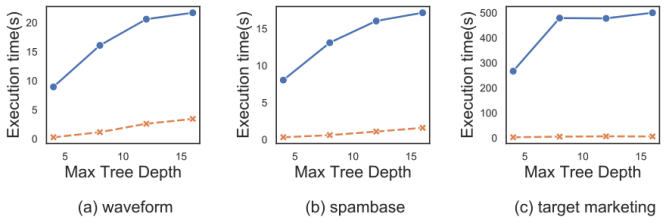


Fig. 7. Prediction Time vs. Max Depth

Finally, we fixed the number of estimators and maximum tree depth, and changed the test sample rate from 0.1 to 0.4, as shown in Figure 8. Because the classical approach has a strong linear correlation with the sample size, we found that its results presented a linear growth trend. Meanwhile, the execution time of our method changed very slowly, which shows our method is robust to the prediction sample size.

Overall, our new prediction method had been proved to be highly efficient.

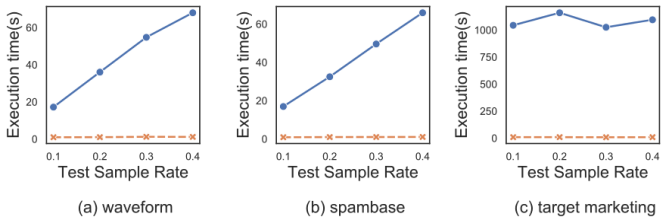


Fig. 8. Prediction Time vs. Test Sample Size

6 Conclusions

In this paper, we proposed a novel tree-based machine learning model, called *Federated Forest*, which is lossless with respect to the model accuracy and protects data privacy. A secure cross-regional machine learning system was developed based on it, which allows a learning model to be jointly trained across different clients with the same user samples but different attribute sets. The raw data on each client are not exposed and exchanged to other clients during the modeling. A novel prediction algorithm was proposed which could largely reduce the communication overhead and improve the prediction efficiency. Data privacy was secured by redesigning the tree algorithms, deploying encryption methods and establishing a third-party trusted server. Raw data will never be directly exchanged, only a limited amount of intermediate values between each party. We performed experiments on both real-world and UCI data sets, showing the superior performance in classification and regression tasks, and the proposed Federated Forest was proven to be as accurate as of the non-federated random forest that requires gathering the data into one place. The convergence, efficiency and robustness of our proposed system have also been verified. Overall, the Federated Forest overcomes the challenges of the *data islands* problem and privacy protection in a brand new approach, and it can be deployed for real-world applications.

Acknowledgement

This work was supported by the National Key R&D Program of China (2019YFB2101805).

References

[1] G. D. P. Regulation, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46," *Official Journal of the European Union (OJ)*, vol. 59, no. 1-88, p. 294, 2016.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[3] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016.

[4] J. Konecný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *CoRR*, vol. abs/1610.02527, 2016.

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

[6] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug 1996.

[7] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *CoRR*, vol. abs/1802.07876, 2018.

[8] V. Smith, C. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017, pp. 4424–4434.

[9] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, "Loadaboost: Loss-based adaboost federated machine learning on medical data," *CoRR*, vol. abs/1811.12629, 2018.

[10] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *CoRR*, vol. abs/1711.10677, 2017.

[11] S. Caldas, P. Wu, T. Li, J. Konecný, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," *CoRR*, vol. abs/1812.01097, 2018.

[12] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 12, 2019.

[13] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," *CoRR*, vol. abs/1812.03337, 2018.

[14] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "Secureboost: A lossless federated learning framework," *CoRR*, vol. abs/1901.08755, 2019.

[15] H. H. Zhuo, W. Feng, Q. Xu, Q. Yang, and Y. Lin, "Federated reinforcement learning," *CoRR*, vol. abs/1901.08277, 2019.

[16] F. M. L. W. Group, "P3652.1 - guide for architectural framework and application of federated machine learning," Available at https://standards.ieee.org/project/3652_1.html, 2019.

[17] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12.

[18] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation, Academia Press*, pp. 169–179, 1978.

[19] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *CoRR*, vol. abs/1712.07557, 2017.

[20] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[21] T. P. Le, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics & Security*, vol. PP, no. 99, pp. 1–1, 2018.

[22] S. Kim, M. Omori, T. Hayashi, T. Omori, L. Wang, and S. Ozawa, "Privacy-preserving naive bayes classification using fully homomorphic encryption," in *Neural Information Processing*, L. Cheng, A. C. S. Leung, and S. Ozawa, Eds. Cham: Springer International Publishing, 2018, pp. 349–358.

[23] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang, "Secure logistic regression based on homomorphic encryption: Design and evaluation," *JMIR medical informatics*, vol. 6, no. 2, p. e19, 2018.

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

[25] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 439–450.

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

[27] J. Vaidya, B. Shafiq, W. Fan, D. Mehmood, and D. Lorenzi, "A random decision tree framework for privacy-preserving data mining," *IEEE transactions on dependable and secure computing*, vol. 11, no. 5, pp. 399–411, 2013.

[28] I. Giacomelli, S. Jha, R. Kleiman, D. Page, and K. Yoon, "Privacy-preserving collaborative prediction using random forests," *AMIA Summits on Translational Science Proceedings*, vol. 2019, p. 248, 2019.

[29] R. Nock, S. Hardy, W. Henecka, H. Ivey-Law, G. Patrini, G. Smith, and B. Thorne, "Entity resolution and federated learning get a federated resolution," *CoRR*, vol. abs/1803.04035, 2018.

[30] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml