Federated Forest

Yang Liu[1,2] **, Yingting Liu**[3,*]**, Zhijie Liu**[4,*]**, Junbo Zhang**[1,2,†]**, Chuishi Meng**[1,2] **, Yu Zheng**[1,2]
[1]JD Intelligent Cities Research, Beijing, China
[2]JD Intelligent Cities Business Unit, Beijing, China
[3]University of Science and Technology of China, Hefei, China
[4]Beijing Normal University, Beijing, China
{liuyang21cn,yingting6,msjunbozhang,msyuzheng}@outlook.com
201621210044@mail.bnu.edu.cn, chuishimeng@gmail.com

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 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.

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 highly 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) [26]. 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. Faced with the difficulties and restrictions, the question becomes if it is worth investing effort to make use of the scattered data.

[*]Equal contribution. The research was done when the second and third authors were interns at JD Intelligent Cities Research.

[†]Junbo Zhang is the corresponding author.

Preprint. Under review.

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 a precised customer profiling and improve their marketing strategies. Government organizations could work with ride-hailing companies to have a better understanding of city's daily traffic flow and adjust the road planing based on it.

Consequently, the question becomes how can we train the joint-models. Faced with the challenges of *data islands* and *data privacy & security*, the current 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 [23, 20, 19] was proposed, which mainly focuses on building privacy-preserved machine learning models when data are distributed in different places. Federated learning has provided a new approach to look at the current problems, and shwon the possibility of real-life applications.

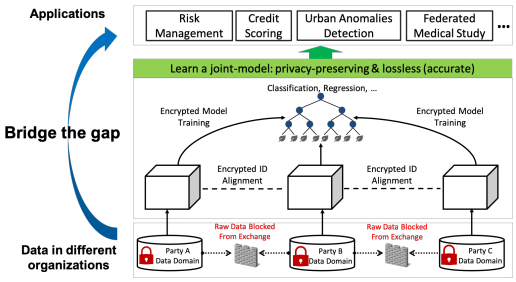

Figure 1: New Era of Machine Learning

Inspired by their work, we proposed 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. 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 [3] and bagging [2], 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 with methods of *MPI* [1] for sharing of the intermediate 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

2.1 Federated Learning

Federated learning [23, 20, 19] 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 [5] they proposed a new recommender system which applies federated learning to meta-learning. Federated learning has also been applied to solve multi-task problems in [28] and a loss-based AdaBoost method was developed in [16]. [14] 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 the data privacy and keep modeling accurate. In addition, a modular benchmarking framework for federated settings was presented in the work of [4]. Although many research products have been coming out, the definition of federated learning was still blurry until the work of [29]. 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 [22] to build models for target-domain

party by leveraging rich labels from source-domain party, as the data sets of the two parties are different in both sample space and feature space. In [6] 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 [30] 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* [12] 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 [8] and homomorphic encryption [10]. The idea of differential privacy is to add properly calibrated noise to the algorithm or the data, with examples including [11, 24]. 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 [15, 21, 18] all used this approach. 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-linear functions, such as Sigmoid and Logarithmic function, and approximations are necessary. In the work of [14] they used the Taylor expansion to approximate the Sigmoid function and [17] used least squares method. In theory these approaches could work but in our practice the results were not ideal.

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 $\mathcal{D}_i$, then the overall data domain is $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_M$, where $1 \leq i \leq M$. M is the number of regional domains. We denote the feature space of $\mathcal{D}_i$ as $\mathcal{F}_i$, then the entire feature space $\mathcal{F}$ is $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_M$. During the modeling process, all features' true names were encoded to protect privacy. For any i and j, if $i \neq j$ and $1 \leq i, j \leq M$, then $\mathcal{F}_i \cap \mathcal{F}_j = \varnothing$. 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 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 modeling together, which could be rare. The model design can be totally different for large M. 2) We are not going to talk about the methods of ID alignment since it is another research topic, discussed in work such as [25]. The notations appeared in this paper are also shown in Table 3.

3.2 Problem Statement

The formal statement of the problem is given as below:

Given: Regional domain $\mathcal{D}_i$ and encrypted label y on each client i, $1 \leq i \leq M$.

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, $1 \leq i \leq M$.

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

4 Methodology

Here we present the framework of Federated Forest, which is based on the CART tree [3] and bagging [2], and is able to deal with both classification and regression problems. The framework is shown as in Figure 2 and details of the algorithms are given in the following subsections.

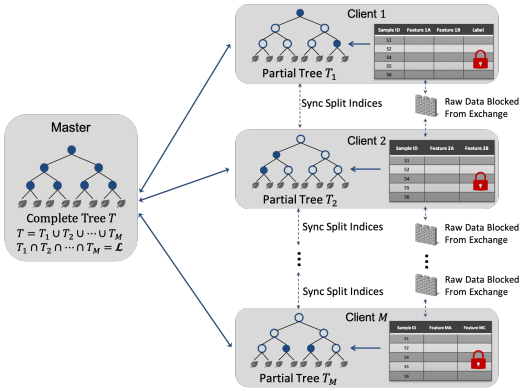

Figure 2: Federated Forest

4.1 Model Building

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 their 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 the selected features and sample IDs privately. For the selected features, 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 break point.

ALGORITHM 1: Federated Forest – Client

Input : Data set $\mathcal{D}_i$ on client i;
Local features $\mathcal{F}_i = \emptyset$ or $\mathcal{F}_i = \{f_A, f_B, \cdots\}$;
Encrypted label y;
Output : Partial Federated Forest Model on Client i
while *tree_build is True* **do**
  Receive $\mathcal{F}_i^{'} \subset \mathcal{F}_i$ and $\mathcal{D}_i^{'} \subset \mathcal{D}_i$ for current tree building;
  Function `TreeBuild` $(\mathcal{D}_i^{'}, \mathcal{F}_i^{'}, y)$
    Create empty tree node;
    if *the pre-pruning condition is satisfied* **then**
      Mark current node as leaf node;
      Assign leaf label by voting;
      return *leaf node*;
    $p, f^* \leftarrow -\infty, None$;
    if $\mathcal{F}_i^{'} \neq \emptyset$ **then**
      Compute impurity improvement p for any $f \in \mathcal{F}_i^{'}$ and find local maximum p_i;
      Record local best split feature f^* and split threshold;
    Send encrypted p_i to master;
    if *receive the split message from master* **then**
      /* Global best split feature is from itself */
      is_selected $\leftarrow$ True;
      Split samples and send sample indices of left and right subtrees to master;
    else
      Receive sample indices of left and right subtrees;
    left_subtree $\leftarrow$ `TreeBuild` $(\mathcal{D}_{i\_left}^{'}, \mathcal{F}_i^{'}, y_{left})$;
    right_subtree $\leftarrow$ `TreeBuild` $(\mathcal{D}_{i\_right}^{'}, \mathcal{F}_i^{'}, y_{right})$;
    if *is_selected is True* **then**
      Save f^* and split threshold to tree node;
    Save subtrees to tree node;
    return *tree node;*
  Append current tree to forest;
return *Partial Federated Forest Model on Client i;*

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 or may not store the detailed information. Only the master server is able to 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 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 $T_1 \cap T_2 \cap \cdots \cap T_M = \mathcal{L}$, where $\mathcal{L}$ is the leaf node sets. The complete tree T is the union of all partial trees, that $T = T_1 \cup T_2 \cup \cdots \cup T_M$.

ALGORITHM 2: Federated Forest – Master

Input : Indices of $\mathcal{D}$;
Encoded features $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_M$;
Encrypted label y;
Output : Complete Federated Forest Model
/*Build trees for forest recurrently*/
while *tree_build is True* **do**
  Broadcast randomly selected samples $\mathcal{D}^{'}$;
  Randomly select features $\mathcal{F}_i^{'}$ from $\mathcal{F}_i$ and send to client i;
  Function `TreeBuild` *($\mathcal{D}^{'}$, $\mathcal{F}^{'}$, y)*
    Create empty tree node;
    if *the pre-pruning condition is satisfied* **then**
      Mark current node as leaf node;
      Assign leaf label by voting;
      return *leaf node*;
    Receive encrypted $\{p\}_{i=1}^{M}$ and related information from all clients;
    Take $j = argmax(\{p\}_{i=1}^{M})$ and notify client j;
    Receive split indices from client j and broadcast;
    left_subtree $\leftarrow$ `TreeBuild` ($\mathcal{D}_{left}^{'}, \mathcal{F}^{'}, y_{left}$);
    right_subtree $\leftarrow$ `TreeBuild` ($\mathcal{D}_{right}^{'}, \mathcal{F}^{'}, y_{right}$);
    Save subtrees and split info to tree node;
    return *tree node;*
  Append current tree to forest;
return *Complete Federated Forest Model*;

4.2 Model Prediction

ALGORITHM 3: Federated Forest Prediction – Client

Input : Partial federated forest model saved on ith client;
Encoded features $\mathcal{F}_i$ on ith client;
Test set $\mathcal{D}_i^{test}$ on ith client;
Output : Samples IDs S_i^l of leaf l on T_i, $l \in \mathcal{L}$
while *TreePrediction is True* **do**
  Function `TreePredict` *(T_i, $\mathcal{D}_i^{test}$, $\mathcal{F}_i$)*
    if *is_leaf is True* **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 subtrees;
        left_subtree $\leftarrow$ `TreePredict` (T_{i_left}, $\mathcal{F}_i$, $\mathcal{D}_{i,left}^{test}$);
        right_subtree $\leftarrow$ `TreePredict` (T_{i_right}, $\mathcal{F}_i$, $\mathcal{D}_{i\_right}^{test}$);
      else
        left_subtree $\leftarrow$ `TreePredict` (T_{i_left}, $\mathcal{F}_i$, $\mathcal{D}_i^{test}$);
        right_subtree $\leftarrow$ `TreePredict` (T_{i_right}, $\mathcal{F}_i$, $\mathcal{D}_i^{test}$);
      Return left and right subtrees;
    Send $S_i = \{S_i^1, S_i^2, \cdots, S_i^l, \cdots\}$ to master;

Under the vertical federated setting [29], 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 which takes the 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.

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.

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 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 $l \in \mathcal{L}$. $\mathcal{L}$ is the set of leaf nodes of the tree T_i.

Thirdly, for each leaf $l \in \mathcal{L}$, the master will take the intersection on $\{S_i^l\}_{i=1}^M$, 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:

ALGORITHM 4: Federated Forest Prediction – Master

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

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 $\{S_i^l\}_{i=1}^M$, that $S^l = S_1^l \cap S_2^l \cap \cdots \cap S_M^l$.*

The proof is provided in Appendix 7. 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.

4.3 Privacy Protection

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 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 on 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 the security protection and the 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 indices, 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 local machine. If not, it only stored the structure of the current node. Clients knew nothing about each other including whose features were selected and at which tree nodes. Master can optionally keep a copy of the entire model.

5 Experimental Studies

5.1 Experimental Setup

In this section, we used 9 benchmark data sets, including one real-world data set *target marketing* and 8 public data sets from UCI [7, 27, 9, 13], as shown in Table 1. 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. In our experiments 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. The *target marketing* data set was collected from two totally different domains. One of them was from an e-commerce company and contains 84 features, and the other one was from a bank which provided 11 features. Before modeling all the sensitive information was protected. Three main series of experiments were conducted in this section, including experiments with two data providers, experiments with multiple data providers, and analysis of prediction efficiency. The details of each test are given in the following subsections.

Table 1: Classification and regression experiments

Classification	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 [27]	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**
Regression	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 [13]	19.74	17.49	17.52	15.369 ± 0.118	15.411 ± 0.163	**0.186**

5.2 Experiments with Two-Party Scenarios

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 original data. For *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 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.
- **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, which two parties jointly learn a random forest. Data were kept locally and model was partly stored in each client.

We conducted the experiments on both classification and regression problems, and present the results of accuracy and RMSE in Table 1. 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 imperfect study of the 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 a 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 same to the tree built by aggregating raw data together. Z-Test 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 1. 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 $0.01 \leq$ *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 3, the accuracy of Federated Forest improved consistently. The training execution time was almost linearly with respect to to the number of domains, which is to be expected because all features are 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.

5.4 Prediction Efficiency

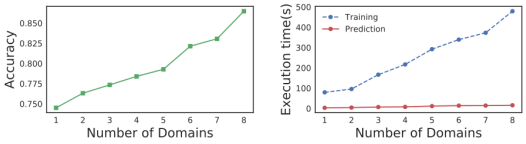

Figure 3: Accy. & Exec. Time vs. # of Domains

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 4, 5 and 6. The solid lines with dot marker represent the results of classical prediction method, and the dash lines with x marker represent our proposed prediction method.

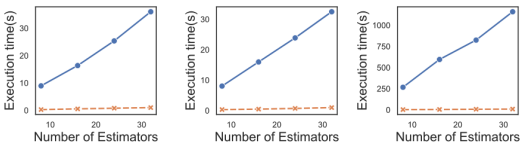

Figure 4: Prediction Time vs. Number of Estimators

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 4. It can be seen that our method produced a strong improvement on the 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.

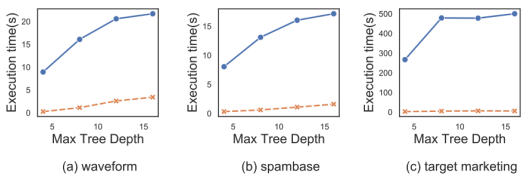

Figure 5: Prediction Time vs. Max Depth

Secondly, we set the number of estimators to 8, and adjusted the maximum tree depth from 4 to 16. As shown in Figure 5, our method outperformed the classical prediction method again. By increasing the maximum tree depth, the 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.

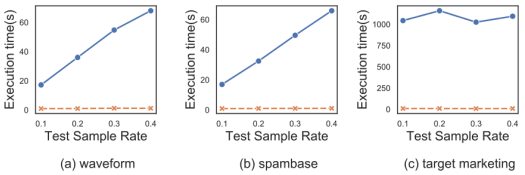

Figure 6: Prediction Time vs. Test Sample Size

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 6. 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 prediction sample size.

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

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 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 the non-federated random forest that requires gathering the data into one place. The 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

Special thanks to Chentian Jin for valuable discussions and feedback.

References

[1] Blaise Barney. 2019. Message Passing Interface (MPI). Lawrence Livermore National Laboratory. Available at `https://computing.llnl.gov/tutorials/mpi`.

[2] Leo Breiman. 1996. Bagging Predictors. *Machine Learning* 24, 2 (01 Aug 1996), 123–140. `https://doi.org/10.1023/A:1018054314350`

[3] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. 1984. *Classification and Regression Trees*. Taylor & Francis.

[4] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. LEAF: A Benchmark for Federated Settings. arXiv:cs.LG/1812.01097

[5] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. 2018. Federated Meta-Learning for Recommendation. arXiv:cs.LG/1802.07876

[6] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, and Qiang Yang. 2019. SecureBoost: A Lossless Federated Learning Framework. arXiv:cs.LG/1901.08755

[7] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml`

[8] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12.

[9] Kelwin Fernandes, Pedro Vinagre, and Paulo Cortez. 2015. A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. In *Progress in Artificial Intelligence*, Francisco Pereira, Penousal Machado, Ernesto Costa, and Amílcar Cardoso (Eds.). Springer International Publishing, Cham, 535–546.

[10] Craig Gentry. 2009. *A fully homomorphic encryption scheme*. Ph.D. Dissertation. Stanford University. `crypto.stanford.edu/craig`.

[11] Robin C. Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially Private Federated Learning: A Client Level Perspective. arXiv:cs.CR/1712.07557

[12] Federated Machine Learning Working Group. 2019. P3652.1 - Guide for Architectural Framework and Application of Federated Machine Learning. Available at `https://standards.ieee.org/project/3652_1.html`.

[13] Kam Hamidieh. 2018. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science* 154 (2018), 346 – 354. `https://doi.org/10.1016/j.commatsci.2018.07.052`

[14] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. 2017. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. arXiv:cs.LG/1711.10677

[15] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. 2017. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. arXiv:cs.LG/1711.10677

[16] Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. 2018. LoAdaBoost:Loss-Based AdaBoost Federated Machine Learning on medical Data. arXiv:cs.LG/1811.12629

[17] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, and Xiaoqian Jiang. 2018. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR medical informatics* 6, 2 (2018), e19.

[18] Sangwook Kim, Masahiro Omori, Takuya Hayashi, Toshiaki Omori, Lihua Wang, and Seiichi Ozawa. 2018. Privacy-Preserving Naive Bayes Classification Using Fully Homomorphic Encryption. In *Neural Information Processing*, Long Cheng, Andrew Chi Sing Leung, and Seiichi Ozawa (Eds.). Springer International Publishing, Cham, 349–358.

[19] Jakub Konecnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. arXiv:cs.LG/1610.02527

[20] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. arXiv:cs.LG/1610.05492

[21] Trieu Phong Le, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. 2018. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. *IEEE Transactions on Information Forensics & Security* PP, 99 (2018), 1–1.

[22] Yang Liu, Tianjian Chen, and Qiang Yang. 2018. Secure Federated Transfer Learning. arXiv:cs.LG/1812.03337

[23] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv:cs.LG/1602.05629

[24] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning Differentially Private Recurrent Language Models. arXiv:cs.LG/1710.06963

[25] Richard Nock, Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Giorgio Patrini, Guillaume Smith, and Brian Thorne. 2018. Entity Resolution and Federated Learning get a Federated Resolution. arXiv:cs.DB/1803.04035

[26] General Data Protection Regulation. 2016. 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)* 59, 1-88 (2016), 294.

[27] C Okan Sakar, Gorkem Serbes, Aysegul Gunduz, Hunkar C Tunc, Hatice Nizam, Betul Erdogdu Sakar, Melih Tutuncu, Tarkan Aydin, M Erdem Isenkul, and Hulya Apaydin. 2019. A comparative analysis of speech signal processing algorithms for Parkinsonś disease classification and the use of the tunable Q-factor wavelet transform. *Applied Soft Computing* 74 (2019), 255–263.

[28] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. 2017. Federated Multi-Task Learning. arXiv:cs.LG/1705.10467

[29] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 12.

[30] Hankz Hankui Zhuo, Wenfeng Feng, Qian Xu, Qiang Yang, and Yufeng Lin. 2019. Federated Reinforcement Learning. arXiv:cs.LG/1901.08277

APPENDIX

Reproducibility

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. The information of all used data sets are given in Table 2.

Table 2: Data sets

Classification	Size	Features	Classes
target marketing	156198	95(11/84)	2
ionosphere	351	34	2
spambase	4601	57	2
parkinson [27]	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 [13]	21263	81	0.0002-185

Pseudo-code for FF-Regressor

The main difference between regression and classification problem lies in the generation of leaf node result and the final predictions. The following is the pseudo-code of regression problem, where the difference from the classification problem is in the line 7 of Algorithm 5, line 9 of Algorithm 6 and line 5 of Algorithm 8.

Notations In Proof

- 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 $\mathcal{H}$, and the single sample is $h \in \mathcal{H}$.
- 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 fall 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, $S^l \cap S^g = \emptyset$.
- The complete tree T on master is defined as $T = T_1 \cup T_2 \cup \cdots \cup T_M$.
- Detailed descriptions of notations are shown in Table 3.

Proof of the Proposition 1

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 $\{S_i^l\}_{i=1}^M$, that $S^l = S_1^l \cap S_2^l \cap \cdots \cap S_M^l$.

Proof. In order to prove $S^l = S_1^l \cap S_2^l \cap \cdots \cap S_M^l$, we will prove:

- $S^l \subseteq S_1^l \cap S_2^l \cap \cdots \cap S_M^l$
- $S^l \supseteq S_1^l \cap S_2^l \cap \cdots \cap S_M^l$

ALGORITHM 5: Federated Forest – Client

Input : Data set $\mathcal{D}_i$ on client i;
Local features $\mathcal{F}_i = \emptyset$ or $\mathcal{F}_i = \{f_A, f_B, \cdots\}$;
Homomorphic encrypted label y;
Output : Partial Federated Forest Model on Client i

while *tree_build is True* **do**
  Receive $\mathcal{F}_i^{'} \subset \mathcal{F}_i$ and $\mathcal{D}_i^{'} \subset \mathcal{D}_i$ for current tree building;
  Function `TreeBuild` *($\mathcal{D}_i^{'}$, $\mathcal{F}_i^{'}$, y)*
    Create empty tree node;
    if *the pre-pruning condition is satisfied* **then**
      Mark current node as leaf node;
      Assign leaf label by averaging;
      return *leaf node*;
    end
    $p, f^* \leftarrow -\infty, None$;
    if $\mathcal{F}_i^{'} \neq \emptyset$ **then**
      Compute impurity improvement p for any $f \in \mathcal{F}_i^{'}$ 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 f^*_{global} is from itself */
      is_selected $\leftarrow$ 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 $\leftarrow$ `TreeBuild` ($\mathcal{D}^{'}_{i\_left}$, $\mathcal{F}_i{}'$, y_{left});
    right_subtree $\leftarrow$ `TreeBuild` ($\mathcal{D}^{'}_{i\_right}$, $\mathcal{F}_i^{'}$, 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;
  return *Partial Federated Forest Model on Client i;*
end

Proof of $S^l \subseteq S_1^l \cap S_2^l \cap \cdots \cap S_M^l$:

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 left and right subtrees simultaneously. Therefore for sample h, its decision making path w^* on the complete tree T must be subset of its decision making path W_i on any client i. Then we have $w^* \subseteq W_i, 1 \leq i \leq M$, which is equivalent to $h \in S_i^l, 1 \leq i \leq M$. Because of this we can safely say that $h \in S_1^l \cap S_2^l \cap \cdots \cap S_M^l$ for any h in S^l. Then we can prove that $S^l \subseteq S_1^l \cap S_2^l \cap \cdots \cap S_M^l$.

Proof of $S^l \supseteq S_1^l \cap S_2^l \cap \cdots \cap S_M^l$:

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 $h \in S_1^l \cap S_2^l \cap \cdots \cap S_M^l$.

$\Longrightarrow h \in S_1^g \cap S_2^g \cap \cdots \cap S_M^g$, obtained by the above proof.

ALGORITHM 6: Federated Forest – Master

Input : Indices of $\mathcal{D}$;
Encoded features $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots \cup \mathcal{F}_M$;
Encrypted label y;

Output : Complete Federated Forest Model

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

ALGORITHM 7: Federated Forest Prediction – Client

Input : Partial federated forest model saved on ith client;
Encoded features $\mathcal{F}_i$ on ith client;
Test set $\mathcal{D}_i^{test}$ on ith client;

Output : Samples IDs S_i^l of leaf l on T_i, $l \in \mathcal{L}$

while *TreePrediction is True* **do**
  Function `TreePredict` *(T_i, $\mathcal{D}_i^{test}$, $\mathcal{F}_i$)*
    if *is_leaf is True* **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 subtrees based on threshold;
        left_subtree $\leftarrow$ `TreePredict` (T_{i_left}, $\mathcal{F}_i$, $\mathcal{D}_{i\_left}^{test}$);
        right_subtree $\leftarrow$ `TreePredict` (T_{i_right}, $\mathcal{F}_i$, $\mathcal{D}_{i\_right}^{test}$);
      else
        left_subtree $\leftarrow$ `TreePredict` (T_{i_left}, $\mathcal{F}_i$, $\mathcal{D}_i^{test}$);
        right_subtree $\leftarrow$ `TreePredict` (T_{i_right}, $\mathcal{F}_i$, $\mathcal{D}_i^{test}$);
      end
      Return left and right subtrees;
    end
    Send $S_i = \{S_i^1, S_i^2, \cdots, S_i^l, \cdots\}$ to master;
  end
  return;
end

ALGORITHM 8: Federated Forest Prediction – Master

Input : Sample IDs S of test set $\mathcal{D}^{test}$;
Output : Prediction of Random Forest

while *TreePrediction is True* **do**
  Gather $\{S_1, S_2, \cdots, S_i, \cdots\}$;
  Obtain $\{S^1, S^2, \cdots, S^l, \cdots\}$, where $S^l = S_1^l \cap S_2^l \cap \cdots \cap S_M^l$;
  Return label of leaf l for samples in S^l, $l \in \mathcal{L}$;
end
Calculate forest predictions by averaging the results of trees;
return *Final Predictions;*

Table 3: Notations

Notation	Description
M	number of regional domains
$\mathcal{D}_i$	data set held by client 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
T_i	partial decision/regression tree stored on ith client
T	complete tree $T = T_1 \cup T_2 \cup \cdots \cup T_M$
$\mathcal{L}$	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

$\Longrightarrow h \in (S_1^g \cap S_2^g \cap \cdots \cap S_M^g) \cap (S_1^l \cap S_2^l \cap \cdots \cap S_M^l)$

$\Longrightarrow h \in (S_1^g \cap S_1^l) \cap (S_2^g \cap S_2^l) \cap \cdots \cap (S_M^g \cap S_M^l)$

That is to say, sample h will fall into the leaf node g and l at the same time in every model stored on 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 $w^l \cup w^j$. And the lowest common ancestor node exists and is uniquely set to O.

So $(w^l \cup w^j) \subseteq W_i \Longrightarrow (w^l \cup w^j) \in (W_1 \cap W_2 \cap \cdots \cap W_M)$

So no platform stores the information of the node O.

$\Longrightarrow T \neq T_1 \cup T_2 \cup \cdots \cup T_M$

This contradicts to $T = T_1 \cup T_2 \cup \cdots \cup T_M$.

Therefor the hypothesis doesn't hold.

$\Longrightarrow h \notin S^l \Longrightarrow h \notin S_1^l \cap S_2^l \cap \cdots \cap S_M^l$

$\Longrightarrow S^l \supseteq S_1^l \cap S_2^l \cap \cdots \cap S_M^l$

In summary, we can prove $S^l = S_1^l \cap S_2^l \cap \cdots \cap S_M^l$.

□

Communication Complexity Analysis

Here we give a brief analysis on 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 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 $O(M)$.