Cross-Silo Federated Learning based Decision Trees

Saikishore Kalloori
Media Technology Center
ETH Zürich, Switzerland
ssaikishore@ethz.ch

Severin Klingler
Media Technology Center
ETH Zürich, Switzerland
severin.klingler@inf.ethz.ch

ABSTRACT

Most research and application in the field of Machine Learning focus on training a model for a particular task such as churn prediction by using training data present on one machine or in a data center. Nowadays, in many organizations and industries, the training data exists in different (isolated) locations. In order to protect data privacy and security, it is not feasible to gather all the training data to one location and perform a centralized training of machine learning models. Federated Learning (FL) is a form of machine learning technique where the goal is to learn a high-quality model trained across multiple clients (such as mobile devices) or data centers without ever exchanging their training data. Most of the existing research on FL focuses on two directions: (a) training parametric models such as neural networks and (b) mainly focusing on an FL setup containing millions of clients. However, in this work, we focus on non-parametric models such as decision trees, and more specifically, we build decision trees using federated learning and train random forest model. Our work aims at involving corporate companies instead of mobile devices in the federated learning process. We consider a setting where a small number of organizations or industry companies collaboratively build machine learning models without exchanging their privately held large data sets. We designed a federated decision tree-based random forest algorithm using FL and conducted our experiments using different datasets. Our results demonstrate that each participating corporate company have benefit in improving their model's performance from federated learning. We also introduce how to incorporate differential privacy into our decision tree-based random forest algorithm.

CCS CONCEPTS

• **Security and privacy** → *Privacy-preserving protocols*; • **Computing methodologies** → *Classification and regression trees.*

KEYWORDS

Federated learning, Random Forest, Decision Tree

ACM Reference Format:
Saikishore Kalloori and Severin Klingler. 2022. Cross-Silo Federated Learning based Decision Trees. In *The 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22), April 25–29, 2022, Virtual Event,* . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3477314.3507149

1 INTRODUCTION

Standard machine learning approaches require centralizing the training data on one machine or in a data center. Recently, a new distributed machine learning approach, called federated learning (FL) enables model training on a large corpus of decentralized data residing in different isolated locations. In a federated learning setup, typically, there is one global server and many clients participate in solving a specific machine learning problem and learn a single shared model. Each client trains the model based on their local private data, and the model updates are sent back to the server. The server aggregates these models updates from all the clients (typically by averaging) to construct an improved model. [15]. The idea of federated learning was introduced by Google [13, 15, 19] to enable mobile phones to collaboratively learn a shared prediction model while keeping all the training data on device [9].

Federated Learning can be categorized into three types [26]. The first is horizontal federated learning introduced in scenarios where the clients have same features but different set of user samples. The second one is vertical federated learning introduced in scenarios where the clients have the same set of users across all the clients but differ in the feature space, i.e, each client has different features. The third one is federated transfer learning where two clients differ in users samples but also in feature space.

Most of the previous research on FL mainly focused towards (a) training parametric models such as neural network [11] under federated learning and (b) training the federated systems designed for millions of real-world mobile devices (each mobile device correspond to single user data). However non-parametric models such as decision trees are widely known as powerful prediction model since they are easy to implement and train [22]. Furthermore federated learning for scenarios where a small number of corporate clients (e.g, media companies such as news publishers) training a machine learning model is still unexplored.

In our work, we propose a non-parametric model for horizontal federated learning and we design a federated random forest [3, 16] which is based on decision trees. Moreover, we consider scenarios where the clients are a small number of organisations or industries collaborating to learn machine learning models [12]. In our scenarios the number of clients are very small and the amount of data present with each client is very large compared to the millions of mobile devices participating the typical federated learning setup [13].

In our proposed algorithm, we use histograms to summarise the data present with each client. The global server constructs decision trees layer by layer by aggregating the histograms of all the clients. Additionally, we also incorporate privacy on histograms while constructing the decision trees to ensure privacy protection. We conducted our experiments on two publicly available datasets and on two datasets from news publishers to show that our proposed federated random forest algorithm can achieve better performance when compared to individual clients' random forest models that are built using only the individual privately held data.

Our contributions can be highlighted as follows:

- We proposed how to build non-parametric model for horizontal federated learning scenarios. In particular, we introduced a novel method of constructing decision trees in federated learning.
- We proposed a random forest algorithm for horizontal federated learning scenarios. In particular our technique of constructing decisions trees can be easily extended for other machine learning models such as AdaBoost [6] or XGBoost [1] for a federated learning scenarios. Additionally, we also introduced privacy into our proposed algorithm.
- We conducted experiments on public and real datasets to show the usefulness of our proposed method in a simulated federated learning setup. We have also integrated our proposed solutions into a publicly available industry level framework [23] for corporate clients to train federated models.

The rest of this paper is structured as follows. In the next section we present the related work and this is followed by illustrating the proposed method of constructing decision trees using federated learning. We next present the description of the evaluation strategy used in our experiments and a comprehensive discussion of the obtained results. Finally, we formulate our conclusions and discuss future work.

2 RELATED WORK

Machine Learning models require access to a large amount of data to build a robust model. The data is likely to be privacy sensitive, which excludes the possibility of sending and storing all the data centrally. Privacy and protecting the data is an increasingly important aspect of the machine learning process. The federated optimization which was introduced by Google [14] has set the path for a new and promising field in machine learning called Federated Learning. Previous research focuses on building scalable federated systems designed for millions of real-world mobile devices [14] and to provide efficient communications [18] across the clients and server. Despite FL being a promising technique, there was no major attention or work done in its application towards training federated models for corporate companies and business organizations [23].

Deep neural networks are widely used for federated learning setup [20], however recently it is also extended to other machine learning models such as support vector machines [24], logistic regression model [10] and extreme gradient boosting [2, 17]. Recently [16] proposed a federated forest that focuses on building decision trees for vertical federated settings, and in this work, we proposed a method focused on horizontal federated settings.

However, in scenarios where corporate companies or industry partners participate in FL, there are various key differences. Firstly, the amount of data i.e., the number of users records and the number of available features vary largely. And the number of clients participating in FL is very few (in our experiments we have 2 to 10). Secondly, the number of communication rounds and sampling the clients for training differ significantly when compared to the traditional FL with mobile devices. It is worth noting that each corporate company already possesses enough training information to build a machine learning model [12] and the major goal for a corporate client to participate in FL is to have a model which performs better than a model that is trained using the client's local private data. In this work, we have very few clients such as industrial companies, and proposed a federated learning-based decision tree algorithm.

In federated learning, there are two widely adopted methods for data privacy and security. They are differential privacy [4] and homomorphic encryption (secure multi-party computation) [7]. Differential privacy has been used widely in machine learning tasks against privacy inference attacks. Many research works focus mainly on adding perturbations on parameters in the gradient descent algorithms [25]. In this work, we apply differential privacy during the training process of decision tree construction.

3 THE PROPOSED METHOD

Random forest (RF) is an ensemble method based on decision (classification or regression) trees and building decision trees is the major function of building a random forest. In this work, we build decision trees for a horizontal federated learning setup. In our scenario, we have a few clients who hold the data for training and a global server that builds a single federated model using all the clients' data. Traditional decision tree-based algorithms use a recursive process to create a decision tree from a training data set. To build decision trees, we make use of histograms which are constructed on each client and then sent to the global server. To identify the best split feature and feature value, the global server aggregates multiple histograms from the clients into a single histogram and calculates the best split value for the feature.

A histogram represents the distribution of the data and contains bins (sometimes also referred to as buckets) that divide the entire range of data values into a series of non-overlapping intervals [21]. Each bin represents the number of data samples that fall into the bin interval. One of our main motivations to use histograms is because they are widely known to be simple to interpret and can be very useful to speed up the training process and reduce memory consumption. During the communication between the clients and global server, histograms can be considered as lightweight for communication and one can construct or merge two histograms in a linear time. In the following paragraphs, we will explain in detail how we explore histograms for building decision trees.

During the learning process, the global server acts as a monitor and coordinates the clients to send the summary of data samples using histograms. The global server that has histograms from all the clients builds a decision tree layer by layer. The overall communication between the global server and clients continues until the random forest is built (i.e. until the learning process is over). The major steps in the learning process are (a) constructing a histogram, (b) merging all the histograms on the global server, and (c) identifying the split value. We will explain how to build a decision tree layer by layer using histograms and we build a random forest simply by building multiple such decision trees.

As the initial step, we ask each client to bootstrap their data. Using the bootstrapped data, we will build a decision tree layer by layer. When the tree is empty, each feature becomes a potential root node and we need to find the best root node. At this step, each client, using all the features, build one histogram per feature and send them to the global server. The global server aggregates all the histograms of each feature and uses evaluation criteria (e.g., entropy or Gini Index) to identify the best split feature and feature value. The global servers then communicate the best split feature and feature value as the root node to all the clients. For simplicity, let say all the data passing through the root node is partitioned into a left node with the left subset of data and a right node with the right subset of data. To build the next layer for the tree, each client again builds a histogram per feature present at the left node and also a histogram per feature present at the right node and then sends them to the global server. The global server aggregates all the histograms per feature coming from the left node and all the histograms per feature coming from the right node. The global server again using evaluation criteria (e.g., entropy or Gini Index), finds the best feature and feature value to split on the left node and also on the right node. This way from the root node, we can build the next layer and this procedure continues till we reach the maximum depth of the tree. Algorithm 1 describes the communication happening between a server and some set of clients. We stress that in horizontal federated learning all the clients have common features and therefore, it is straightforward to merge the histograms coming from the same feature from different clients.

Algorithm 1: Communication Between Server and Clients

```
Input: Set of clients participating in the federated learning
Output: A random forest model
1 Initialize a random forest $F$ as an empty set on global server
2 while The number of trees in the random forest is not achieved do
3     Repeat the following steps until the decision tree is fully developed
          • Each client constructs a histogram per feature and sends it to the server.
          • The server merges all the histograms and identifies best feature and its best split value.
```

Mathematically, let us assume that we have a global server G and a set of n clients $C = \{C_1, C_2, ..., C_{n-1}, C_n\}$ who possess data. Let us also denote the set of features that are same among all the clients as $\mathcal{F} = \{f_i\}_{i=1}^{d}$. All the clients are interested to collaboratively train and learn a random forest algorithm (machine learning model). Given a data $D = \{(x_1, y_1), ..., (x_m, y_m)\}$ where $x_m = (f_1, ..., f_d)$ is the input feature vector of size d and y_m is the output value for the input vector x_m. For simplicity, let the label values be either 1 (positive label) or 0 (negative label). In our scenarios, when a histogram is built for a feature f_i we have two cases as follows: (i) If the feature contains discrete values (categorical values), then the number of bins for the histogram is equal to number of discrete values (ii) if the feature contains continuous values then, a predefined number of bins is used for constructing a histogram. We will explain how to construct a histogram when the features have continuous values and it is straight forward to construct the histogram for discrete values.

As mentioned above, each client bootstrap data samples as $S \subset D$. We will construct a decision tree using the bootstrapped data S. Note that each time a new decision tree is to be constructed, the data is bootstrapped again. The reason to bootstrap the data is to construct different decision trees. For a feature f_i, we define a histogram h_{f_i} over a bootstrapped data S as a set B containing b number of bins summarising the data samples of S corresponding to the feature f_i:

$$h_{f_i} = \{(r_1, p_1, n_1), ..., (r_b, p_b, n_b)\} \tag{1}$$

where b is a predefined constant integer. Each bin $B_i \in B$ is a tuple (r_i, p_i, n_i) summarizing a small set of data samples S_{B_i} with p_i as the number of positively labeled data points, n_i as the number of negatively labeled data points and r_i as the bin identifier for B_i. The value of r_i is the average of feature values of the data point that fall into S_{B_i} as

$$r_i = \frac{1}{p_i + n_i} \sum_{(x_i, y_i) \in S_{B_i}} x_i(f_i) \tag{2}$$

where $x_i(f_i)$ is the value of feature f_i of the input feature vector x_i. We stress that the histogram h_{f_i} can be built over a data set S in a single pass and for each data sample $(x_i, y_i) \in S$, we define a tuple:

$$(x_i(f_i), p_i, n_i) = \begin{cases} (x_i(f_i), 1, 0) & \text{If } y_i \text{ is a positive label} \\ (x_i(f_i), 0, 1) & \text{If } y_i \text{ is a negative label} \end{cases} \tag{3}$$

During the construction, there can be scenarios where the number of bins in the histogram exceed the predefined maximum value b. Therefore, we merge two nearest bins present in the histogram. The indices for nearest bins can be calculated using the following equation:

$$i_{min}, j_{min} = \underset{i,j \in (1,...,b)}{\operatorname{argmin}} |r_i - r_j| \tag{4}$$

and they can be replaced by a merged bin B_{merge} obtained from two bins $B_{i_{min}}, B_{j_{min}}$ using the following equation:

$$B_{merge} = \left(\frac{(p_{i_{min}} + n_{i_{min}}) * r_{i_{min}} + (p_{j_{min}} + n_{j_{min}}) * r_{j_{min}}}{p_{i_{min}} + p_{j_{min}} + n_{i_{min}} + n_{j_{min}}}, p_{i_{min}} + p_{j_{min}}, n_{i_{min}} + n_{j_{min}} \right) \tag{5}$$

Algorithm 2 describes the construction of a set of histograms $\mathcal{H} = \{H_{f_i}\}_{f_i \in \mathcal{F}}$ for a given set of feature $\mathcal{F}$ and with b as predefined number of bins. Having discussed how to construct histograms for a given set of features in a single pass over a data sample S, the next step is to construct one histogram as H_{merge} on the global server from all the histograms collected from the clients. We will now explain how to merge multiple histograms $\{H_{f_i}^c\}_{c=1}^{n}$ corresponding to the same feature f_i from various clients. Algorithm 3 describes the merge procedure on the global server that create one histogram by the union of all the clients histograms. Once the global server finishes the histogram H_{merge}, the next step is to identify how to split the feature. In order to identify an optimal split decision for a given feature in the decision tree, one requires a set of candidate

Algorithm 2: Histogram construction for feature set $\mathcal{F}$ on a client $C_i \in C$

```
Input: Data set $S$, number of bins $b$ and feature set $\mathcal{F}$
Output: Set of Histograms where each histogram
        corresponds to each feature $f_i \in \mathcal{F}$
1  For all $f_i \in \mathcal{F}$, initialise a histogram $h_{f_i} = \{\}$
2  for $(x_i, y_i) \in S$ do
3      Iterate over each feature $f_i \in \mathcal{F}$
4      Construct the current bin $B_i = (r_i, p_i, n_i)$ using
         equation 3
5      if $r_i = r_j$ for some $B_j = (r_j, p_j, n_j) \in h_{f_i}$ then
6          $p_j = p_i + p_j$
7          $n_j = n_i + n_j$
8      else
9          add $(r_i, p_i, n_i)$ to histogram $h_{f_i}$
10     if the size of histogram $|h_{f_i}| > b$ then
           • Find two nearest bins $B_i, B_j$ using
             the equation 4
           • Construct the merged bin $B_{merge}$ using
             equation 5
           • Update the set of bins present in histogram
             $h_{f_i}$ as: $h_{f_i} \leftarrow (h_{f_i} \setminus \{B_i, B_j\}) \bigcup \{B_{merge}\}$
```

threshold values. Using H_{merge}, we propose to use the mid points between two adjacent bins $B_i, B_j \in H_{merge}$ as candidate threshold values. We first sort the bins of H_{merge} according to bin-identifiers, i.e, r_i values of bins (r_i, p_i, n_i). We then calculate mid-points as $(r_i + r_j)/2$ between adjacent bins B_i, B_j and evaluate these points in terms of some metric, usually information gain or Gini impurity.

We note that we assumed that the feature has continuous values however when the feature has discrete values, the construction and merging of histograms are straightforward (the number of bins is the number of discrete values and r_i is the discrete value of the corresponding bin) since we only look at each discrete value the feature posses and the number of bins will equal to the number of discrete values.

3.1 Differentially Private Histograms

In this section, we present the mechanism to introduce Differential Privacy (DP) into the histograms which are shared between the clients and global server. In the above tree construction process, we add noise to histogram bins to make them differentially private. DP was initially proposed by [5] and the underlying principle is that the output of a computation should not allow inference about any particular record/sample in the input. This is achieved by requiring that the probability of any computation outcome is insensitive to small input changes. Laplace mechanism is one of the widely used methods to obtain ϵ−differential privacy on the continuous data by directly adding noise sampled from a Laplace distribution. For categorical data, the exponential mechanism is used which changes the value of a categorical feature by uniformly sampling from other possible categorical values of the feature. The parameter ϵ directly controls the privacy guarantees. Having $\epsilon = 0$ implies

Algorithm 3: Histogram merging for feature variable f_i on the global server G

```
Input: Multiple histograms $\left\{H^c_{f_i}\right\}_{c=1}^{n}$ from all the $n$ clients
       and number of bins $b$
Output: Histogram $H_{merge} = \{(r_1, p_1, n_1), ......, (r_b, p_b, n_b)\}$
1  Initialise a histogram $H_{merge} = \{\}$
2  for each $H^c_{f_i} \in \left\{H^c_{f_i}\right\}_{c=1}^{n}$ do
       • Add all the bins of $H^c_{f_i}$ to $H_{merge}$
       • if $r_i = r_j$ for some $B_j = (r_j, p_j, n_j) \in H_{merge}$ then
           $p_j = p_i + p_j$
           $n_j = n_i + n_j$
3  Sort the histogram bins in $H_{merge}$ according to their
     bin-identifiers.
4  while the size of histogram $|H_{merge}| > b$ do
       • Find the two nearest bins $B_i, B_j \in H_{merge}$ using the
         equation 4
       • Construct the merged bin $B_{merge}$ using equation 5
       • Update set of bins present in histogram $H_{merge}$ as:
         $H_{merge} \leftarrow (H_{merge} \setminus \{B_i, B_j\}) \bigcup \{B_{merge}\}$
```

perfect privacy guarantees, but a very bad utility of the data. In order to make the histogram ϵ−differentially private, we apply geometric mechanism [8] and add independent noise for each bin present in the histogram. The geometric mechanism, for a given input value $f(d)$, it outputs $f(d) + Z$ where Z is a random variable. The random variable Z is distributed as a two-sided distribution with every value having the following probability:

$$P[Z = z] = \left(\frac{1-\epsilon}{1+\epsilon}\right) * \epsilon^{|z|} \tag{6}$$

This mechanism is ϵ-differentially private because the probabilities of adjacent points in its range differ by an ϵ factor.

4 EXPERIMENTS

In order to investigate the performance of our proposed algorithm, we performed experiments using various datasets and simulated a federated learning setup. We distributed the data among a set of clients and collaboratively train a global model without collecting data onto the server. This section describes how we have built our experimental setup and datasets to investigate the performance of our proposed algorithm.

Dataset: We used two publicly available datasets and two real datasets from local news publishers of Switzerland. Among the public datasets, the first one is Telco churn prediction[1] dataset and it contains 7043 user samples described by 21 features. The second one is KKbox[2] from KDD cup challenge. Moreover, we study the performance of our proposed algorithm using real datasets from two news publishers (referred as publisher A and publisher B) of Switzerland. There are total of 18 features that are common across both the publishers describing each user. Publisher A contains 7000

[1]https://www.kaggle.com/blastchar/telco-customer-churn
[2]https://www.kaggle.com/c/kkbox-churn-prediction-challenge

user samples and publisher B contains 20,000 user samples. All the datasets correspond to two-class classification problems.

Evaluation Scheme: For our experiments, to set up clients and data for each client, we vary the number of clients participating in the federated learning and we divide a given (public) dataset into roughly equal-sized subsets in a stratified fashion; thus, we preserve the label distribution across the clients. We first fix the number of clients $n = 2, 5, 10$ participating in the federated learning and for each n we split the dataset into n subsets. Furthermore, every client then splits up his data into a test set and a training set. Each client uses 80% of the data as training data, and the test data contains the remaining 20% of the data. The split into training and test data is done in a stratified fashion to ensure that both the training data and the test data have equal class label distribution. From the training data, each client uses 20% of the local training data as validation data for tuning the hyperparameters. In our experiments, we used F1 and AUC metrics to evaluate the performance of our algorithm. We repeat the above procedure five times with randomly splitting data, and the experimental results are averages of five runs.

Baseline Algorithm It is worth noting that each corporate client already possesses enough training information to build a random forest model and the major goal for a corporate client to participate in FL is to have a model which performs better than a model that is trained using their local private data alone. If the federated model does not perform better than the baseline model, then it does not bring any benefit for clients to participate in federated learning. Therefore, we include *local model* as the baseline model to compare against the federated model. A local model is the RF model built by each client's training data alone without participating in the federated learning setup. This is a strong baseline to understand the gain or loss in accuracy achieved from federated learning for corporate clients. We note that a local model can be any machine learning model, but since we are training a decision tree using federated learning, we considered random forest as our local model algorithm.

5 EVALUATION RESULTS

5.0.1 Performance Comparison of Federated Model. In order to understand the usefulness of our federated learning-based random forest algorithm, we compared the performance of our federated model (trained using all the clients' data in a federated learning setup) against a local client's random forest model trained using only the privately held data of the respective client. Figures 1, 2 and 3 present the results of F1 score by comparing the performance of client model against federated model. We can note that our results demonstrate that the federated model is able to achieve a significantly better F1 score than a model that is trained using the client data alone (local model) for all the different client setups. We observed for clients C2, C4, C9, C10 federated random forest is significant ($p < 0.001$) better than their respective local random forest models. We note that our proposed model is considered for scenarios where the clients are corporate companies having multiple users data. In traditional federated learning, it is not feasible to construct a decision tree using single-user data (such as mobile device data). Our proposed technique of tree construction can be easily applied to other boosting algorithms as AdaBoost [6] and will be very useful for domains such as health informatics where each hospital (considered as a client) can build classification models by collaborating with other hospitals. Our results also demonstrate that non-parametric models can be used in a federated learning setup. We believe that such learning of machine learning models across different organizations/industries can be very beneficial for many complex machine learning problems such as recommender systems or health informatics.

Figure 4 shows the performance of the federated model against the local model under two clients setup using the KKbox dataset. We also performed our experiments under five clients setup, however, we do not report those results since the results are similar to two clients setup. For KKbox dataset also, our results show that our federated random forest model significantly ($p < 0.05$) outperformed local random forest models.

Figure 5 shows the results obtained when training the federated model from two news publishers. We note that using publicly available datasets, we have synthetically created the clients' setup. However, with the two publishers dataset, we trained a federated random forest model under a two-client setup by considering each publisher as a single client. Similar to other datasets, our federated model is able to achieve better performance compared to each locally built random forest model. We observed that for publisher A, there is a significant improvement ($p < 0.05$) against publisher A's local model. Overall our results demonstrate that our proposed random forest model[3] can be efficiently used to train models for corporate companies.

5.0.2 Effects of Differential Privacy. In order to secure the preservation of data, we explore two possible approaches to add differentially private noise. We conducted experiments by adding differentially private noise either to the data directly before training the random forest or by adding to histogram during the learning process as follows:

- Input perturbation: The data of each client is first perturbed with a calibrated noise (such as Laplace mechanism) and the federated random forest is trained from the noisy input data from each client i.e, the input perturbation is performed before training the model.
- In-process mechanisms: In this approach, we add noise to histograms which are exchanged between clients and global server as described in the section 3.1. This way we incorporate differential privacy during training the model.

In the experiments, we explored different ϵ values to understand the effect on the model performance. The different ϵ values are $(0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 1.0, 3.0, 5.0, 100.0)$. Figures 6 and 7 show the effect of adding noise either directly to data before training or during training to histograms for two client setups. For each client present in the two-client setup, we observed similar results, therefore we report the results obtained for client C1. We note that in the figures higher values mean lower noise is added and for all the metrics higher score means better model performance. We observe that when we add noise to data, the model performance significantly drops when more noise is added whereas when we

[3]We have integrated our codes into federated learning framework and can be readily used by any corporate companies:https://github.com/MTC-ETH/Federated-Learning-source

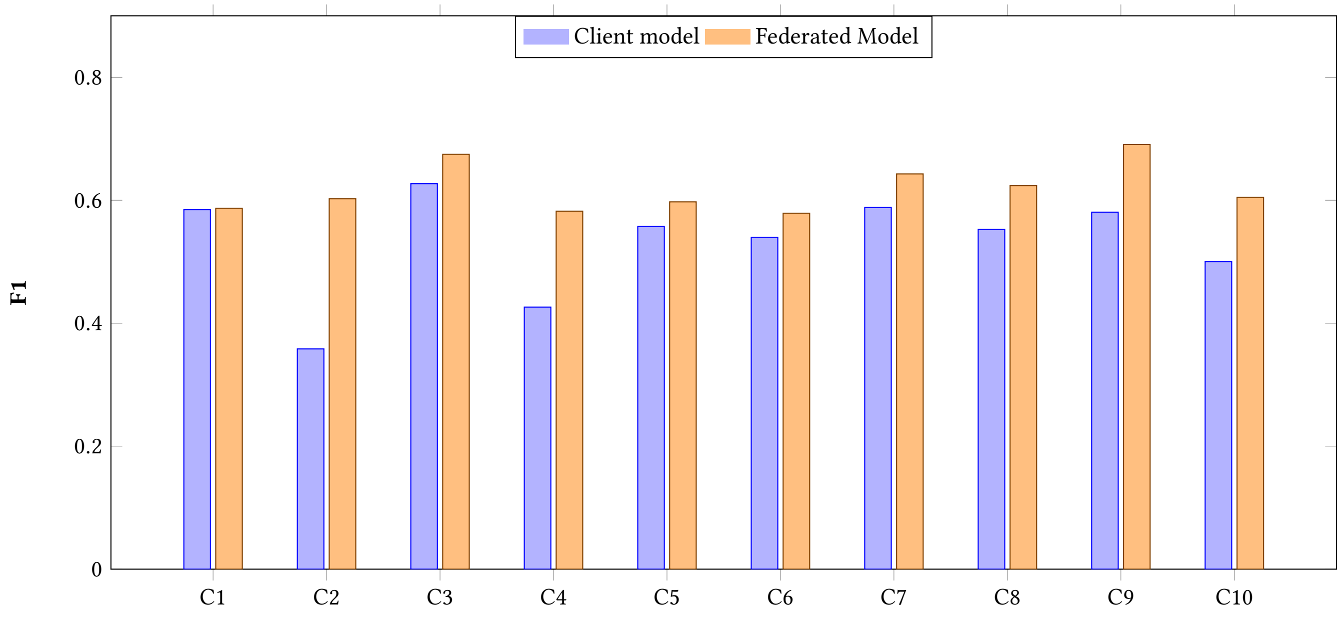

Figure 1: Results for Ten Clients Setup on Telco Dataset

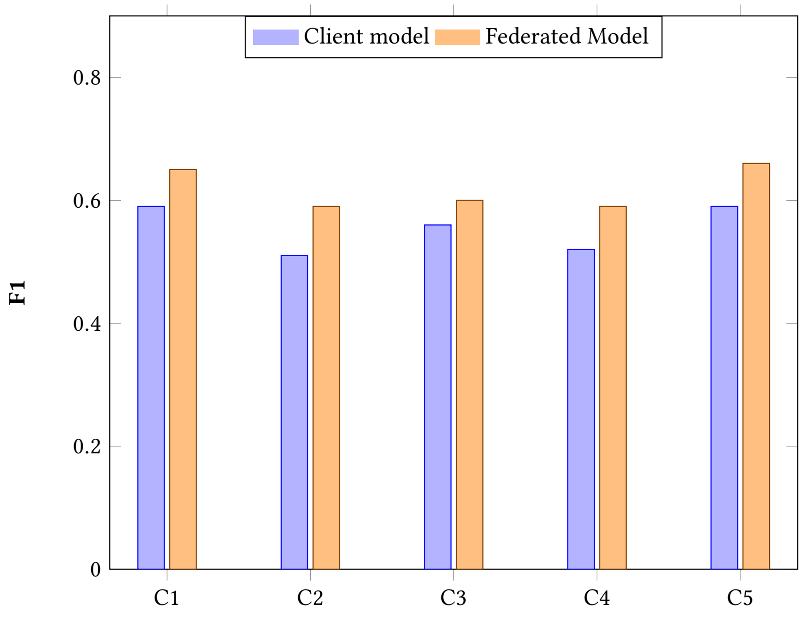

Figure 2: Results for Five Clients Setup on Telco Dataset

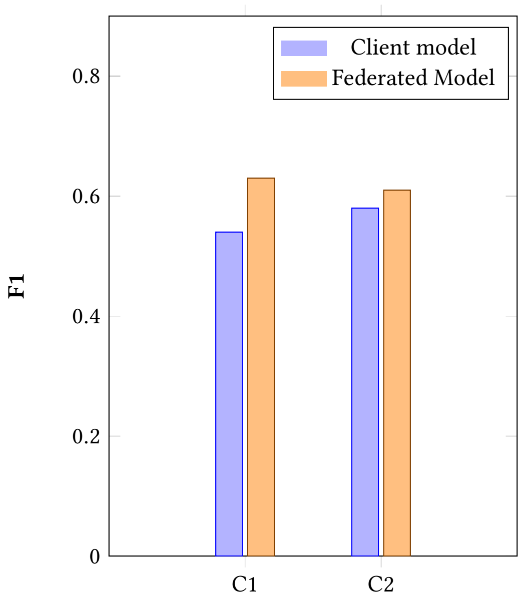

Figure 3: Results for Two Clients Setup on Telco Dataset

add noise to histograms the model is stable, and the performance of the model drops at epsilon = 0.01. Our results demonstrate that adding noise to histograms during the training will lead to good model performance while ensuring high privacy.

5.0.3 Effect of number of bins in histogram. During the model training, we use a pre-defined number of bins b for histogram construction. As mentioned before, for features with categorical values, the number of bins will be equal to the number of distinct categorical values. Thus, the number of bins will have an effect for features with numerical values. Different values of b lead to different patterns in the histogram, and consequently different model performance. Figure 8 shows the performance of the model when we vary the number of bins defined to construct histograms for a two-client setup, for the fixed, best, value of the other meta-parameter. We observe that a low number would lead to low model performance and for our data-set, we noticed that a clear increase in the model performance up to a certain number (in our case

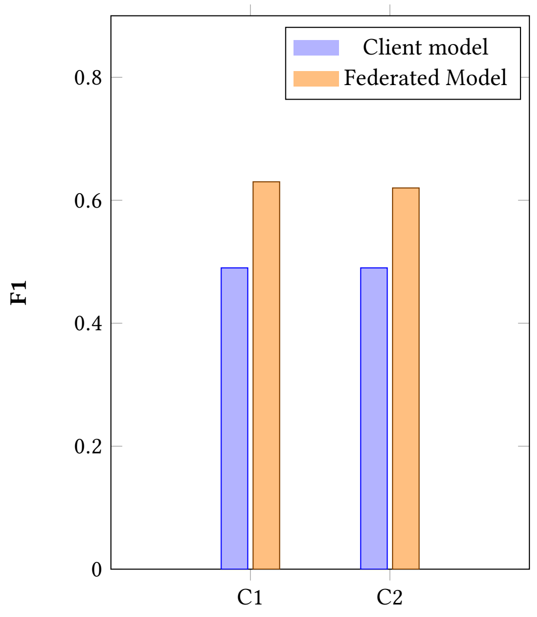

Figure 4: Results for two clients setup on KKbox

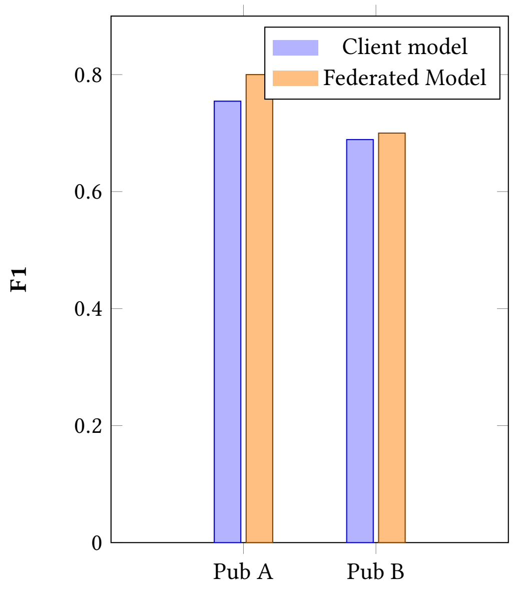

Figure 5: Results for two clients setup on publishers data

8), and thereafter the performance does not have any significant improvement for both the clients.

6 CONCLUSION AND FUTURE WORK

In this work, we focused on non-parametric models i.e., decisions trees for federated learning. We designed the construction of the decision tree using histograms for a horizontal federated learning setup. Our work is focused on industry companies/organizations to collaboratively build machine learning models. Our results show

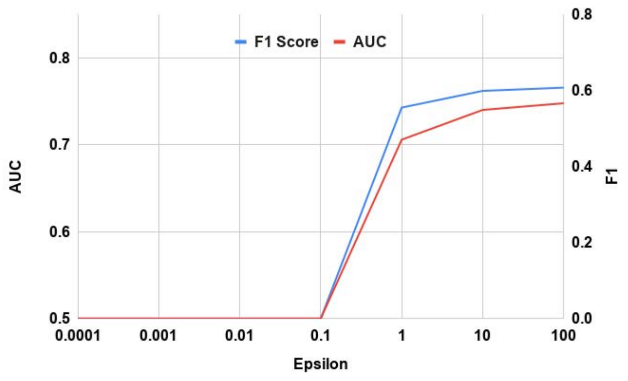

Figure 6: The effect of input perturbation on client C1

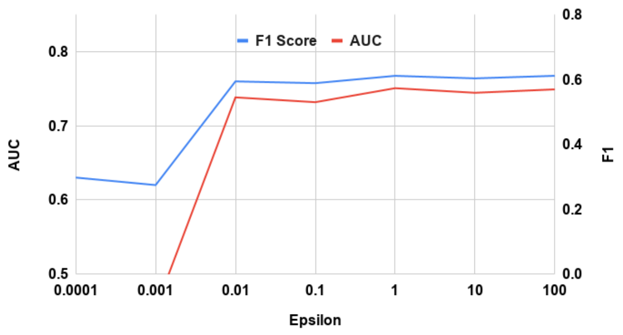

Figure 7: The effect of in-process mechanisms on client C1

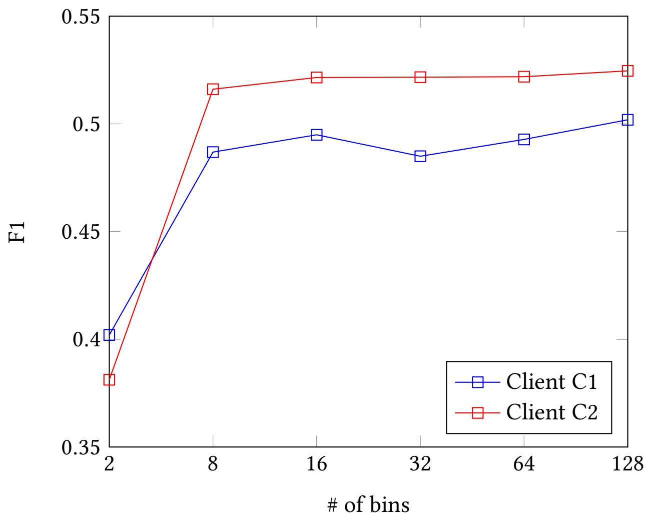

Figure 8: The effect of number of bins used to construct histogram for two client setup

that our proposed federated model can be useful for horizontal federated learning setup and by adding differential privacy into histograms, it is possible to have privacy preservation and protection of the data. In our future work, we would like to explore designing and constructing decision trees for more complex machine learning problems like ranking and recommendations.

7 ACKNOWLEDGEMENT

This project is supported by Ringier, TX Group, NZZ, SRG, VSM, viscom, and the ETH Zurich Foundation. We would also like to thank Christian Schneebeli for the useful discussion and running some of our experiments.

REFERENCES

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

[2] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, and Qiang Yang. 2019. Secureboost: A lossless federated learning framework. *arXiv preprint arXiv:1901.08755* (2019).

[3] Lucas Airam C de Souza, Gabriel Antonio F Rebello, Gustavo F Camilo, Lucas CB Guimarães, and Otto Carlos MB Duarte. 2020. DFedForest: Decentralized Federated Forest. In *2020 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 90–97.

[4] Cynthia Dwork. 2006. Differential privacy, in automata, languages and programming. *ser. Lecture Notes in Computer Scienc* 4052 (2006), 112.

[5] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[6] Yoav Freund, Robert E Schapire, et al. 1996. Experiments with a new boosting algorithm. In *icml*, Vol. 96. Citeseer, 148–156.

[7] Craig Gentry and Dan Boneh. 2009. *A fully homomorphic encryption scheme*. Vol. 20. Stanford university Stanford.

[8] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. 2012. Universally utility-maximizing privacy mechanisms. *SIAM J. Comput.* 41, 6 (2012), 1673–1693.

[9] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[10] 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 preprint arXiv:1711.10677* (2017).

[11] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).

[12] Saikishore Kalloori and Severin Klingler. 2021. Horizontal Cross-Silo Federated Recommender Systems. In *Fifteenth ACM Conference on Recommender Systems*. 680–684.

[13] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).

[14] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).

[15] 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 preprint arXiv:1610.05492* (2016).

[16] Yang Liu, Yingting Liu, Zhijie Liu, Yuxuan Liang, Chuishi Meng, Junbo Zhang, and Yu Zheng. 2020. Federated forest. *IEEE Transactions on Big Data* (2020).

[17] Yang Liu, Zhuo Ma, Ximeng Liu, Siqi Ma, Surya Nepal, and Robert Deng. 2019. Boosting privately: Privacy-preserving federated extreme boosting for mobile crowdsensing. *arXiv preprint arXiv:1907.10218* (2019).

[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. 1273–1282.

[19] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Aguera y Arcas. 2016. Federated learning of deep networks using model averaging. (2016).

[20] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963* (2017).

[21] Karl Pearson. 1895. X. Contributions to the mathematical theory of evolution.—II. Skew variation in homogeneous material. *Philosophical Transactions of the Royal Society of London.(A.)* 186 (1895), 343–414.

[22] Charles S Roehrig. 1988. Conditions for identification in nonparametric and parametric models. *Econometrica: Journal of the Econometric Society* (1988), 433–447.

[23] Christian Schneebeli, Saikishore Kalloori, and Severin Klingler. 2021. A Practical Federated Learning Framework for Small Number of Stakeholders. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 910–913.

[24] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. 2017. Federated multi-task learning. In *Advances in Neural Information Processing Systems*. 4424–4434.

[25] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 245–248.

[26] 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), 1–19.