Membership Inference Attacks on Federated Horizontal Gradient Boosted Decision Trees

Jaap Meerhof
Rijksinstituut voor Volksgezondheid en Milieu
Strategic Program RIVM
University of Twente
Enschede, Netherlands
jaapmeerhof@pm.me

***Abstract*—Federated Learning is often presented as a privacy preserving measure, as the raw unprocessed data is not transferred to other parties. Instead models or gradients for example are shared around such that the original data is hidden. The privacy claims of Federated Learning have been called into question after successful privacy breaching attacks on the model or protocol. By attacking the model or entire federated protocol itself Membership Inference Attacks could retrieve if an individual was present in a dataset. These attacks are especially dangerous in the medical domain; where sensitive data require privacy guarantees. If these attacks can be mitigated in a Federated Learning scenario, medical institutions could collaborate to create machine learning models together. Much of the research is oriented towards Neural Networks, whereas the medical sector and government institutions often use more classical machine learning methods due to their interpretability and ease of use. Attacking Federated Learning Gradient Boosted Decision Trees algorithms is a field mostly left unexplored, therefore this paper investigates the Horizontal Federated Learning protocol "FederBoost" with XGBoost's regularisation parameters. FederBoost is investigated by attacking with two different methods that use extra information acquired during the Federated Learning process. This is done after testing the susceptibility of Gradient Boosted Decision Trees to the Membership Inference Attack without this federated information. This is all done to asses to what extent Gradient Boosted Decision Trees preserve privacy when using Federated Learning with and without heavy encryption methods. One of the two methods that used the leaked federated information was successful and improved the accuracy of the Membership Inference Attack in certain conditions, thus showing the danger of sharing gradients and hessians during training.**

***Index Terms*—Machine Learning, Classical Machine Learning, Privacy, Membership Inference, Federated Learning**

I. Introduction

fData mining is becoming more important and prevalent every year. Machine learning methods require as much high quality data as possible. This data collection and exploitation can be both to the benefit as to the detriment to the individual. Some of the most private information is medical data, which is at the same time highly useful data. Medical data could for example help a machine learning algorithm do early detection of a treatable disease. The easiest way to train such a machine learning algorithm is to collect all data at one central point. This might, however, be near impossible to do with sensitive data which cannot be shared easily between parties without legal contracts and a cyber secure environment. Federated Learning (FL) has been proposed to train a model without sharing the training data [1]–[3]. Federated Learning could help to create strong models by combining datasets from different parties, while not sharing the personal data with other (untrusted) parties. Federated Learning could allow users to train one model without hurting the accuracy of the end model with other privacy preserving measures.

Medical institutions have an interest in creating more accurate and privacy-preserving machine learning models. This is difficult to do as health data holds a higher degree of regulation [4]. Federated Learning could provide both privacy and usability, as other privacy preserving measures often require either slow encryption algorithms or add noise to the data or outputs resulting in a less accurate model.

Research has, however, shown that Federated Learning could introduce new privacy challenges as information about the original dataset can be retrieved from the machine learning models themselves. These attacks include the Membership Inference [5], Property Inference [6], and the Model Inversion [7] Attack. These attacks could still force potential users of Federated Learning to use costly privacy preserving defenses like differential privacy and secure aggregation.

Most of the research into the privacy of federated networks is however going into networks that use Neural Networks. Both in the research on attacking machine learning models as in defending against these attacks. The popularity of Neural Networks is in good reason. These Neural Networks are powerful and have enabled many new technologies, however, they are also more difficult to interpret and use than the "classical" machine learning algorithms. While there is research going into the explainability of Neural Network schemes [8], [9], classical machine learning approaches like Decision Trees, Support Vector Machines, K-Nearest Neighbors, and Random Forest (to a lesser certain extent) have this property already. This property, together with their ease of use, and other properties makes it such that many data scientist are still using classical machine learning methods like Logistic Regression, Decision Trees, Random Forest, and Gradient Boosted Decision Trees like XGBoost. This was found when surveying 25,000 thousand data scientist users on the machine learning website Kaggle [10]. The fact that difficult to interpret models can be unacceptable in the medical sector [8], together with the fast convergence of classical approaches and competitive accuracies [11], creates a strong argument for some to use these classical machine learning approaches. There are papers that provide different ways to do the three different attacks [12]–[16]. The interpretability of Gradient Boosted Decision Trees is questionable, however, when necessary trees could be investigated to asses why a certain result was given for an individual.

There are some papers [13], [16]–[18] that take the information leakage of a federated algorithm into account. All these papers look at Neural Network algorithms however. There is no research that uses the extra leaked information that can be obtained during a Federated Learning process for an attack on Gradient Boosted Decision Trees. Together with this the papers that look into attacking classical machine learning algorithms are limited.

Then there are also papers that apply these different attacks, however they often do not look at applying a defensive measure [6], [16], [17], [19]–[21]. Also some papers have mixed results with their attacks, as shown in Truex and de Arcaute [14], [20]. Most of these papers look at the Neural Networks. This together with the findings of Song and Mittal [22] that shows that different defences might not be as strong as reported in some papers. The effect of regularisation in these federated networks as a defensive measure is not looked at deeply.

To add to this disparity in research, some attacks differ for Horizontal Federated Learning, Vertical Federated Learning, and Federated Transfer Learning. This paper investigates the Horizontal Federated Learning approach as it is the approach most likely to be used in the near future by the Dutch RIVM (National Institute for Public Health and the Environment). In this setting the different participants hold different rows/items/patients. However these differing rows contain the same features. Thus different hospitals could keep track of the age, height, and weight of different patients, however, no hospital is holding a different extra feature.

This research will therefore look into Horizontal Federated Gradient Boosted Decision Trees, and in particular XGBoost's regularisation parameters. XGBoost is a popular method as it incorporates extra regularization, approximation, it is sparsity aware, and offers many computational optimisations to create an overall package that can rival Neural Networks in certain settings. The extra regularization of XGBoost is of particular importance as it could help defend against possible attacks.

This paper analyses the following Main Research Question (MQR):

- **MRQ**: What level of impact does taking away regularisation, and secure aggregation have on the privacy of individuals when using Horizontal Federated Gradient Boosted Decision Trees?

To achieve this the following Research Questions (RQ) will have to be answered:

- **RQ1**: To what extent do the gamma, alpha, lambda, learning rate, max depth, the number of trees, the number of buckets, and training size regularisation parameters affect the effectiveness of the Salem *et al.* Membership Inference Attack when attacking Gradient Boosted Decision Trees?
- **RQ2**: To what extent can communicated non-secure-aggregated differentials in Horizontal Gradient Boosted Decision Tree be used in a membership inference attack?

To answer this question GBDT will be attacked using the Membership Inference Attack on a Horizontal Federated Learning network from Tian *et al.* [23] called FederBoost. To defend against the Membership Inference Attack different levels of regularisation will be tested and compared. The extra information that could leak if secure aggregation is not used will be exploited to attack FederBoost. This paper thus introduces new methods of doing a Membership Inference Attack.

In Section II the different terms/technologies, abbreviations and notations used throughout the paper are explained. In Section III FederBoost together with its privacy model is defined. Section IV explains the different attacks done against FederBoost. Here the different threat models that apply to the different attacks are also explained. Section V then reveals the results of the different attacks. Section VI goes over other research in the field of privacy breaching attacks against machine learning algorithms. Section VII discusses the results further, and reasons on possible future work. The conclusion to the paper's above mentioned research questions are given in Section VIII.

II. Preliminaries

Abbreviation and notations used throughout the paper can be found in Table I and II respectively.

Abbreviation	Definition
FL	Federated Learning
MLaaS	Machine Learning as a Service
DT	Decision Tree
GBDT	Gradient Boosted Decision Tree
RF	Random Forest
NN	Neural Network
DNN	Deep Neural Network
FCNN	Fully Connected Neural Network
GAN	Generative Adversarial Network
LR	Linear Regression
NB	Naive Bayes
API	Application Programming Interface
PAX	Party Adaptive XGBoost

TABLE I: Abbreviation definitions used in this paper.

Notation	Description
$\mathcal{P}$	set of Participants
l	number of participants
q	number of buckets
T	set of trees
m	number of features
$\mathcal{D}_{pi}$	$\mathcal{P}_i$'s Dataset
y	ground truth label
$\hat{y}$	prediction result
$p(\hat{y})$	probability estimations per class
g	gradient; first order differential of loss function
h	hessian; second order differential of loss function
Q	quantile
C	set of target classes
c	class number c
n	nodeID
k	featureID
S	splits array
v	quantileID
x	data (features)

TABLE II: Notations

A. XGBoost

Chen and Guestrin [24] are the original authors of Xtreme Gradient Boost which is better known as its abbreviation XGBoost; a Gradient Boosted Decision Tree algorithm. In this algorithm Decision Trees are made by iteratively improving on previously made Decision Trees. This process is called "boosting". This boosting process can be started by taking random prediction guesses, on these guesses the first and second order differentials of a loss function can be calculated. The differentials are calculated per person (instance). These differentials can be used to create new trees. All trees are stored throughout this process, to do prediction all trees are traversed to retrieve weights. In this paper only the multi-class problem is considered, to do multi-class prediction with a Gradient Boosted Decision Tree, a set of trees is created for every class. The trees in this set are tasked with predicting if one specific end class is the target y, or if it is not. All tree sets of every class are then collaboratively used to retrieve the probability values of all classes by comparing weights. Thus prediction is done by taking the output of every tree in a set of trees; as can be found in Equation 1. $f_{c,t}$ is the t'th tree out of all T trees for class c. It returns a single weight per tree, which are aggregated to retrieve one weight for every class. These weights are then fed into the softmax function (Equation 2) to retrieve a probability score per class, these probabilities add up to 1 and are in the interval of $[0, 1]$.

$$\hat{y}_{c,i} = \phi(\mathbf{x}_i) = \sum_{t=1}^{T} f_{c,t}(\mathbf{x}_i) \tag{1}$$

$$p[c] = p(\hat{y}_c) = \frac{e^{\hat{y_c}}}{\sum_{c=1}^{C} e^{\hat{y}_c}} \tag{2}$$

Firstly the predictions are set to be random, on these random predictions the first set of differentials can be calculated using Equation 3 and 4. The gradient g being the first order differential and the hessian h being the second order differential.

$$g = \begin{cases} p[c] - 1 & \text{if y == c} \\ p[c] & \text{otherwise} \end{cases} \tag{3}$$

$$h = max(2 * p[c] * (1 - p[c]), 1e - 6) \tag{4}$$

To build the trees, a spit value, split feature pair has to be found at the root and the newly created nodes afterwards. With small simple datasets XGBoost looks at every possible value for the different features and assesses which is best. However, for Federated Learning algorithms quantile sketches are often made to create bins that multiple feature values will fall into. This reduces the amount of splits to asses and with enough bins often barely impacts performance in most scenarios. Quantiles can be sketched with decentralised quantile sketching algorithms like found in the federated XGBoost FATE algorithm [25].

To build these trees iteratively, the dataset is used to find the best split that optimises Equation 5. Here the aggregated gradients and hessians of all instances left (I_L) of a split are compared to the gradients and hessians right of a split (I_R) and all instances (I). These aggregated gradients and hessians are retrieved by adding the gradients or hessians that apply to instances that fall in the same bin. Thus if there is a bin for people aged 0 to 20 then all these people's gradients are aggregated.

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i}\right] \tag{5}$$

When the maximum depth is reached, or the loss returned by Equation 5 is smaller then 0 (can happen when regularisation metrics are introduced as explained in Section II-A1), that node is marked to become a leaf. The weight of a leaf is calculated using Equation 6 using the learning rate ϵ.

$$w = -\frac{\sum_{i\in I} g_i}{\sum_{i\in I} h_i} * \epsilon \tag{6}$$

Eventually no nodes can be made anymore, and thus a tree is made. The weights are then used in Equation 1 and 2 to repeat the cycle.

After the trees are made, other data can be tested upon by predicting on all trees again using Equation 1 and 2.

1) Regularisation Terms: XGBoost uses a lot of different parameters of which most impact the regularisation strength of the end model. The different regularisation metrics investigated in this paper are explained below.

Alpha: α can be in $[0, \infty)$; it functions as the L1 regularisation term. The parameter is used in the split loss score calculation and can be found in Equation 7. α is subtracted from an aggregated gradient g if α is larger g. Else if g is smaller than $-\alpha$, α is added to g. If none of these apply then 0 is returned. This is applied to the summed & squared gradients of Equation 9, before the loss is calculated, and on

Algorithm 1 Approximate & hist algorithm for split finding [24]

```
1: for $k = 1$ to $m$ do
2:     propose split array for feature $k$ $S_k = \{s_{k1}, s_{k2}, ..., s_{kl}\}$ by percentages on feature $k$.
3:     proposal can be done per tree (global approximate), or per split (local approximate) or once in the beginning (hist).
4: end for
5: for $k = 1$ to $m$ do
6:     $G_{kv} \longleftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{j,k} > s_{k,v-1}\}} g_j$
7:     $H_{kv} \longleftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{j,k} > s_{k,v-1}\}} h_j$
8: end for
```

the new weight calculation Equation 8. A larger α will result in a smaller numerator. And thus a smaller loss resulting in a more conservative model.

$$L1(g, \alpha) = \begin{cases} g - \alpha & \text{if } g > \alpha \\ g + \alpha & \text{if } g < -\alpha \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Gamma: γ can be in $[0, \infty)$ and is used to prune the tree. If the calculated loss in Equation 9 is below zero then the split is not taken. Thus γ introduces a minimum value that the loss has to be for a split to be taken.

Lambda: λ can be in $[0, \infty)$ and functions as the L2 regularisation term. Increasing this value will make the model more conservative as the denominator will be larger. As used in the updated Equation 9 and 8.

$$w = -\frac{L1(\sum_{i \in I} g_i, \alpha)}{\sum_{i \in I} h_i + \lambda} * \epsilon \tag{8}$$

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{L1((\sum_{i \in I_L} g_i)^2, \alpha)}{\sum_{i \in I_L} h_i + \lambda} + \frac{L1((\sum_{i \in I_R} g_i)^2, \alpha)}{\sum_{i \in I_R} h_i + \lambda} - \frac{L1((\sum_{i \in I} g_i)^2, \alpha)}{\sum_{i \in I} h_i + \lambda}\right] - \gamma \tag{9}$$

nBuckets: $nBuckets$ can be in $[1, \infty)$ and represents the amount of maximum quantile buckets. With more buckets, the size of every bucket decreases. This will result in quantiles that represent the exact values more and more. Like with the learning rate an epsilon value is often used to notate these quantile sketching algorithms where $\epsilon = \frac{1}{nBuckets}$, in this paper only $nBuckets$ is used to indirectly refer to this privacy parameter epsilon value.

$|\mathbf{T}|$**:** The amount of trees created for every class c; $|T|$ can have an influence on the regularisation. The more trees created, the more dataset is used and information about the original dataset is revealed.

Train_size: The size of the training set has a influence on the regularisation. A small dataset makes it so that the trees are focused on only a small amount of data, making overfitting more likely. A large dataset for Gradient Boosted Decision Trees is more difficult to overfit with Gradient Boosted Decision Trees.

B. Federated Learning

Federated Learning (FL) is a way for a machine learning algorithm to be trained by multiple parties with data while not having to share the data directly. Federated Learning was proposed by Google [1]–[3] who claimed the following: "Federated Learning can significantly reduce privacy and security risks by limiting the attack surface to only the device, rather than the device and the cloud." [1]. This claim is valid, however, due to the broad definition of Federated Learning, sometimes Federated Learning is not that private as may be believed. When using Neural Networks the model could be shared to other users for them to calculate the backpropagations and communicate them back to a central party. This way the model could be trained while keeping data on-premise. The privacy benefits of these protocols are continually being challenged requiring extra privacy preserving measures on top of Federated Learning. Federated Learning can be done with a central party, or distributed without a central party orchestrating the protocol. In this paper only the setting with one central party is investigated.

C. Secure Aggregation

Secure aggregation allows for addition without revealing the underlying addend values used. This can be used to aggregate gradients together like done in Bonawitz *et al.* [26]. Bonawithz *et al.* proposed a secure aggregation protocol to specifically protect the gradients of a Neural Network Federated Learning protocol from Google. Thus only the aggregated sum of the party's addends will be revealed. This algorithm relies on Shamir's t-out-of-n Secret Sharing [27]. Secure aggregation is used in Tian *et al.* 's [23] FederBoost to protect the differentials by aggregating them securely. FederBoost is used in this paper without using this Secure Aggregation.

D. Membership Inference

The Membership Inference Attack is an attack that determines if a data record was used in the training of that model. The attack takes advantage of the fact that machine learning models are often more confident in their assessment when using training data. It does so by creating "shadow" models that aim to be as similar as possible to the actual model. This was first successfully done in a paper from Shokri *et al.* [5]. In this attack one shadow model is made for every class. These models can be trained by creating data using the model itself, creating data using background information, or by having some real noisy data. They show that a Membership Inference Attack can be robust even if the attacker's assumptions about the distribution of the target model's training data are not very accurate. This background information could be retrieved from other open source data; for example from data of another nation. The goal in the end is thus to create a function that given a set of features values, estimates if this set was used in training; like defined in Equation 10. Here $\mathbf{x}$ is a data sample

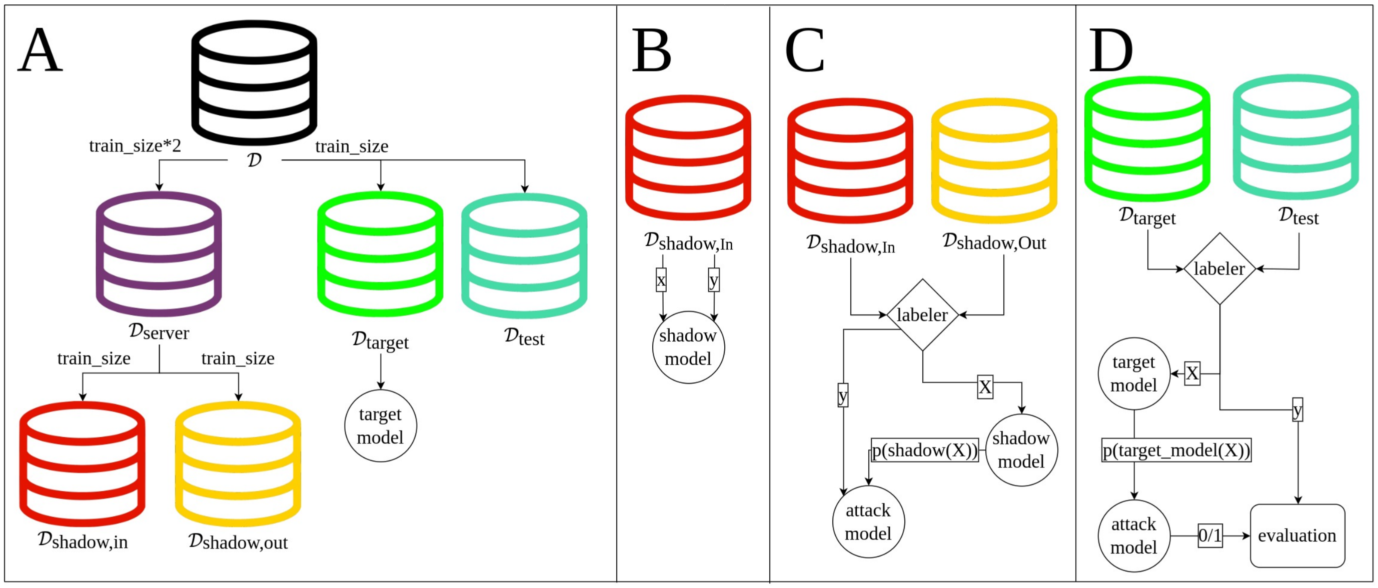


Fig. 1: The Membership Inference Attack as done in Salem *et al.* [15]. **A**: The to be tested dataset $\mathcal{D}$ is split up between $\mathcal{D}_{sever}, \mathcal{D}_{target}$ & $\mathcal{D}_{test}$, where $\mathcal{D}_{target}$ is used to train the target model, the attacking server splits up $\mathcal{D}_{server}$ into $\mathcal{D}_{shadow,in}$ and $\mathcal{D}_{shadow,out}$ both of size $train\_size$. **B**: Server trains the shadow model using $\mathcal{D}_{shadow,in}$. **C**: Server trains the attack model using the trained shadow model. Data elements from both $\mathcal{D}_{shadow,in}$ and $\mathcal{D}_{shadow,out}$ are labeled using Equation 12. **D**: The attack model is evaluated for performance by using $\mathcal{D}_{target}$ and $\mathcal{D}_{test}$ to check if the attack model can differentiate between used data elements and those which were not used.

with the same features as used in training. Function f then returns 0 if $\mathbf{x}$ was used in training and 1 otherwise.

$$\mathcal{A} : \mathbf{x}, f \rightarrow \{0, 1\} \tag{10}$$

This attack was boosted by Salem *et al.* [15] by successfully only using one shadow model instead of multiple achieving similar accuracies. This makes the attack easier to do, thus increasing the potential risk. Salem *et al.* also show that just abusing confidence scores with a threshold-choosing methods can be enough to do an effective Membership Inference Attack. The attack as done in Salem can be found in Figure 1 and is explained below.

Throughout the following explanation a data set is notated as $\mathcal{D}$ and contains the training data X, and the labels/targets y as found in Equation 11.

$$\mathcal{D}_{name} = (X_{name}, y_{name}) \tag{11}$$

A+B: Starting with a dataset $\mathcal{D}$ (in experimentation: “healthcare”, “synthetic-10” or “synthetic-100”) is split up by taking a chunk of size $train\_size * 2$ and $train\_size$. This is distributed to the attacker (server) and the target into $\mathcal{D}_{server}$ and $\mathcal{D}_{target}$ respectively. The target then trains its “target model” with $\mathcal{D}_{target}$. In this paper the dataset $\mathcal{D}_{server}$ and $\mathcal{D}_{target}$ are of the same distribution, this gives the attacker the best case scenario to mimic the target model. The attacker ends splits up its datasets, one will be used to train a shadow model that tries to mimic the target model. As the shadow model is trained in the same way with data similar to the target model, the shadow model is trying to become a “shadow” of the target model. Knowing how the target trained its model might not always be possible, however, in this paper the worst is assumed. The target model is then trained using $\mathcal{D}_{target}$. **C:** The instances used in training $\mathcal{D}_{shadow,in}$’s y values are replaced with ones with the labeler function (Algorithm 2), this indicates that it was used in training. The $\mathcal{D}_{shadow,out}$’s y values are replaced with zeros, this indicates that the entries were not used in training. The untouched instance values $x \in X$ where X represents the combined instances of the $\mathcal{D}_{shadow,in}$ and $\mathcal{D}_{shadow,out}$ are fed into the trained shadow model to retrieve probability scores per instance. These probability scores (one for every class) are used together with y to train the attack model. **D:** To test the attack model, the test dataset $\mathcal{D}_{test}$ is used in combination with the used training samples $\mathcal{D}_{target}$. The labeler function (Algorithm 2 is again used to replace y values as in step **C**, this time the probability scores of the target model are used to do inference on the attack model. The result from the already trained attack model are then used to evaluate how well the attack model is able to differentiate trained instances from unused instances.

When using Federated Learning the Membership Inference Attack can use the messages communicated between the users to try to advance that attack. This is done in Nasr *et al.* [28] for Neural Networks. Thus when the target model trains centrally, the attacker is often only assumed to have access to the target model. In the federated setup, the central party can store the messages send and use them later in an attack. This is illustrated in Figure 2.

Algorithm 2 Labeler function

```
Input: $\mathcal{D}_{out}, \mathcal{D}_{in}$, $take\_h$ (optional)
Output: data elements not used in training are labeled 0, others 1
1: $X$ = the union $\cup$ of $\mathcal{D}_{in}$, $\mathcal{D}_{out}$ such that both there is a 50/50 split between $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$.
2: if $take\_h$ is given then
3:     $X = X.take(random, take\_h)$    # takes random rows
4: end if
5: $y = label(X)$    # uses Equation 12
6: return $(X, y)$    # $= \mathcal{D}_{attack1}$
```

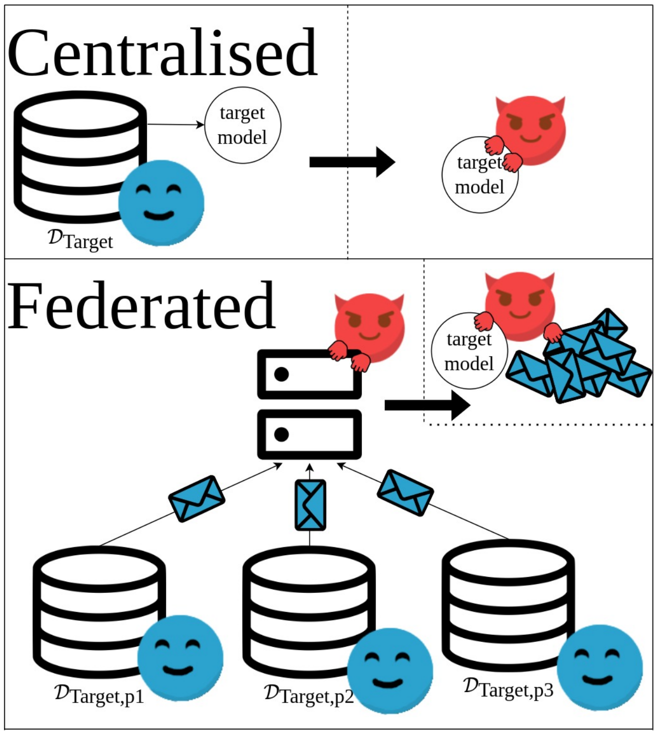

Centralised
target model
$\mathcal{D}_{\text{Target}}$
target model
Federated
target model
$\mathcal{D}_{\text{Target,p1}}$
$\mathcal{D}_{\text{Target,p2}}$
$\mathcal{D}_{\text{Target,p3}}$

Fig. 2: Threat model of both the Centralised and Federated Setting. In the Centralised model, the target model with all its parameters, weights, and splits are available to the attacker. In the Federated setting the attacker is the central coordinating server, and thus also has access to all messages send to it.

III. FederBoost

This section explains the FederBoost algorithm from Tian *et al.* [23]. This algorithm provides an accurate model comparable to running XGBoost centrally with all data. It aims to provide a fast way to create a model by using lightweight secure aggregation. FederBoost was replicated in Python and can be accessed on GitHub.

A. Algorithm

In the horizontal setting the parties will first have to find quantiles for every feature. FederBoost describes it own quantile sketch algorithm; this algorithm uses secure aggregation to hide the information about the dataset of a data holder. However, any (secure) distributed quantile sketch algorithm can be used in place.

1) Participant: A participant $Pi \in P$ has two tasks, first it needs to help find quantiles for the combined dataset among other participants. The quantiles can be found using the aforementioned algorithm described in FederBoost [23] or any other distributed (secure) quantile sketching method. For the testing in this paper DDSketch [29] was used. Secondary, a participant is responsible for calculating first order differentials (gradients) and the second order differentials (hessians) on its data and communicating it with the main server.

Algorithm 3 describes how the different participating user help in creating a federated model. After the quantiles are found the users will initialise the initial predictions to be random (line 2). Using this initial prediction the different users can calculate their first and second differentials (line 5-6); the gradients and hessians respectively; on the dataset using a loss function. These gradients and hessians can be used to create the aggregated gradients and hessians for the different features relative to the instances that are still relevant in the current node. When going down a decision tree, different data points are not being assessed anymore, the data points that are still relevant to a node are the instances I still used in the gradient calculations (line 5-6). The aggregation is done for every class, feature, node and split on the gradients still relevant to that node (line 9-16). These gradients are send to the server, which will receive a two-dimensional array of gradient values for every node n of a tree responsible for class c. The two-dimensional array holds values over the different features and split possibilities for that feature. The gradients are send using secure aggregation such that only the aggregated gradients of all participants can be retrieved by the central party. The server processes the aggregated gradients and sends back the best split found for every node being worked on. The local tree is updated with this split decision on the side of the participant. For every new node the relevant instances I are updated using the split information from the parent's node split information. The prediction probabilities are updated after a tree has been fully constructed.

Algorithm 3 FederBoost (Local Node)

```
Input: $Init\_Probas$, Initial probabilities for every class; $T = \emptyset$, Trees; $\epsilon$, error parameter.
Output: All trees $T = \{t_0, ..., t_{|T|}\}$
 1: run distributed quantile sketch algorithm to retrieve Quantile Splits $S$
 2: initialize $\hat{y}$ randomly
 3: for $t = 1 \rightarrow |T|$ do
 4:   for k = 1 to $m$ do
 5:     $g_k \leftarrow \partial_{\hat{y}} L(y, \hat{y})$
 6:     $h_k \leftarrow \partial^2_{\hat{y}} L(y, \hat{y})$
 7:   end for
 8:   for $d = 1 \rightarrow MAX\_DEPTH$ do
 9:     for $c = 1 \rightarrow NCLASSES$ do
10:       for node $n$ on depth $d$ do
11:         for k = 1 to m do
12:           $G_{cnkv} = \sum_{j \in \{j | s_{kv} \geq x_{jk} > s_{k,v-1}\}} g_k$
13:           $H_{cnkv} = \sum_{j \in \{j | s_{kv} \geq x_{jk} > s_{k,v-1}\}} h_k$
14:         end for
15:       end for
16:     end for
17:     send all $G, H$
18:     receive $split_{cn}$
19:     update node with splits
20:     update instances $I$ of node $n$ in tree for class $c$ based on $split_{cn}$ parent
21:   end for
22:   update $\hat{y} += Predict(f_t)$
23: end for
```

2) Server: The server P_{server} (or Active Party in some papers) is responsible for receiving differentials from the participants and using them to find the best split value and feature combination. This can be done by using Equation 9. The differentials are first received and aggregated using secure aggregation (line 4-9 **Algorithm 4**. The optimal split is then calculated using Equation 9 for every class and node being investigated (line 10-15). The splits are then send to every participant (line 16-15). This is done till all trees have been constructed.

Algorithm 4 FederBoost (Server)

```
Input: $Init\_Probas$, Initial probabilities for every class; $T = \emptyset$, Trees; $\epsilon$, error parameter.
Output: All trees $T = \{t_0, ..., t_{|t|}\}$
 1: receives the chosen quantiles Quantile Splits $S$ from one of the participants
 2: for $t = 1 \rightarrow |T|$ do
 3:   for $l = 1 \rightarrow MAX\_DEPTH$ do
 4:     $G, H \leftarrow \emptyset$
 5:     for $i = 1 \rightarrow |P|$ do
 6:       receive $G_i, H_i$
 7:       $G \cup G_i$
 8:       $H \cup H_i$
 9:     end for
10:     for $c = 1 \rightarrow N\_CLASSES$ do
11:       for node $n$ on level $l$ do
12:         $split_{cn} \leftarrow find\_split(S, G_{cn}, H_{cn})$
13:         As in Algorithm 1.
14:       end for
15:     end for
16:     for $i = 1 \rightarrow l$ do
17:       send all $split_{cn}$
18:     end for
19:   end for
20: end for
```

B. Privacy

Tian *et al.* [23] explains the security achieved by this algorithm in the following manner:

For a participant P_i, there are two places of possible information leakage:

- During quantile lookup, P_i inputs n'_i to secure aggregation
- During the tree construction the gradients and hessians are not leaked as secure aggregation is used.

As secure aggregation is used only the sum of the gradients, hessians and dataset size is leaked.

For P_{server} is also argued that the server leaks some information to the other participants:

- During quantile lookup, $\mathcal{P}_{server}$ sends the quantiles to other users.
- During tree construction the different splits are shared to the participants.

As the splits that are calculated by the server only use the gradients and the hessians, the server does not gain anything from these splits. The deeper the trees are the less users/rows are linked to the gradients & hessians. Thus revealing more about the underlying data. This is illustrated in Figure 3; here the people relevant to a node lower as with larger depth values. By using the differentials leaked at the different nodes, the attacker could extract more information about how the different instances flowed through the tree. This could result in knowing better in which ranges a patients features lie in. The attack from Salem *et al.* [15] only uses the probabilities derived from the end leaf's weights. Although the weights are calculated from the gradients and hessians, some information is lost in this translation. The hypothesis is that the federated attack could benefit from also using the gradients, hessians and tree structure to advance the attack.

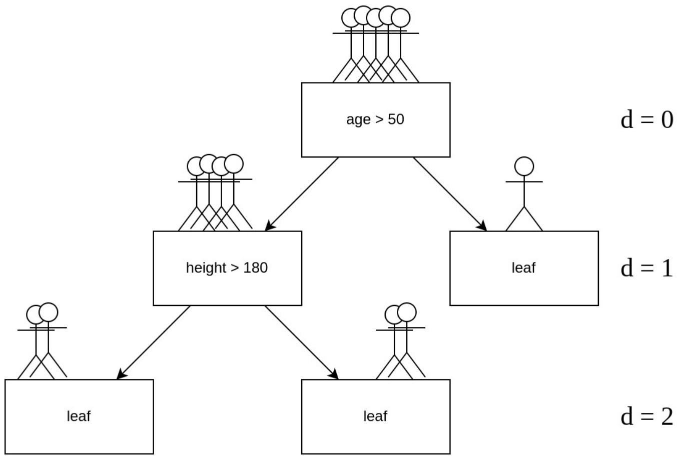


Fig. 3: Deeper trees result in fewer relevant instances down in the tree. The reported gradients are therefore more valuable further down in a tree.

The secure aggregation is used as information about the individual user's dataset can leak to the central server. If an participant reports a gradient of 0 for one of the branching nodes, then that participant has no indices with the feature values, and splits found above towards the root.

IV. Attacking Experiments

This Section will explain what kind of Membership Inference Attacks will be experimented with on the FederBoost algorithm explained in Section III.

A. Environment

In the three experimentation's two threat scenarios will be investigated, the "Centralised" setting and the "Federated" setting. The scenarios are illustrated in Figure 2. In the centralised setting target model is trained by the dataset holder, the attacker gains access to this attack model after training and is tasked with doing the Membership Inference Attack on the target model using background information. In the federated setting the target model is trained in the federated setting. The attacker is the central party coordinating the Federated Learning setup. It can thus capture the different messages send of the different parties and use them later to do the Membership Inference Attack. This means that the gradients and hessians are used, together with the tree split values, and spit features as illustrated in Figure 4C.

The federated attacks are also compared to the centralised attack by using the attack from Salem *et al.* [15] on the federated algorithm (FederBoost-Central).

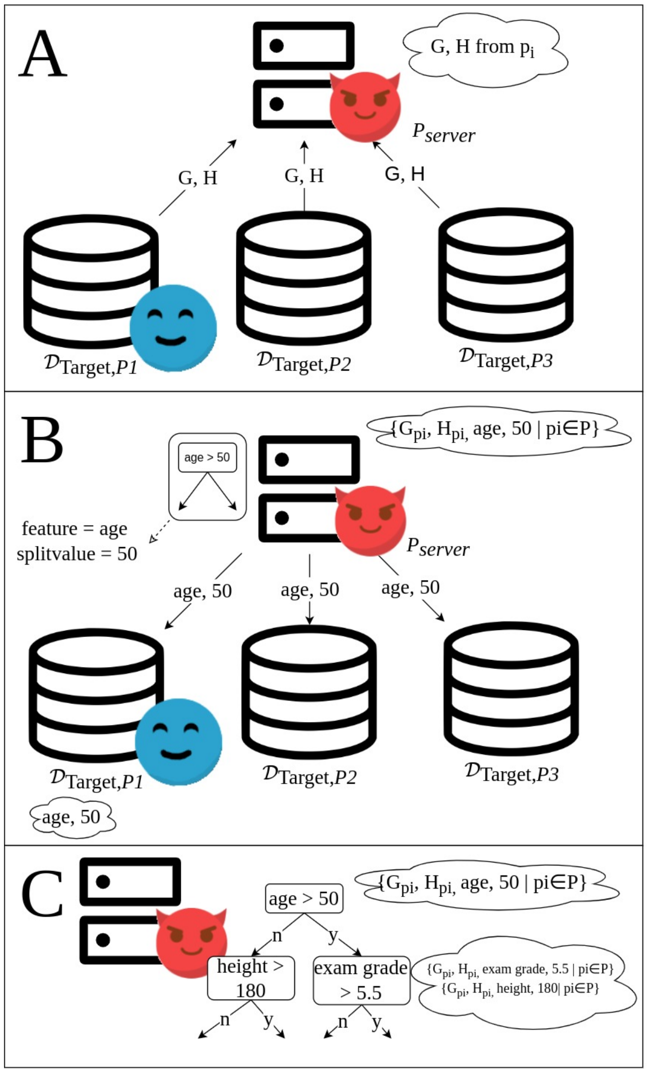

A
G, H from p_i
P_{server}
G, H
G, H
G, H
$\mathcal{D}_{Target,P1}$
$\mathcal{D}_{Target,P2}$
$\mathcal{D}_{Target,P3}$
B
{G_{pi}, H_{pi}, age, 50 | pi∈P}
age > 50
feature = age
splitvalue = 50
P_{server}
age, 50
age, 50
age, 50
$\mathcal{D}_{Target,P1}$
$\mathcal{D}_{Target,P2}$
$\mathcal{D}_{Target,P3}$
age, 50
C
age > 50
{G_{pi}, H_{pi}, age, 50 | pi∈P}
n
y
height > 180
exam grade > 5.5
{G_{pi}, H_{pi}, exam grade, 5.5 | pi∈P}
{G_{pi}, H_{pi}, height, 180| pi∈P}
n
y
n
y

Fig. 4: **A**: During the federated learning process the participants $\{P_1, ...P_i, ..., P_l\}$ will send their aggregated gradients and hessians G_{Pi}, H_{Pi}, these are stored by the central party. **B**: Splits are created, these splits are stored by the central party as they are relevant to following gradients and hessians to be received from lower nodes. The participating parties also store the splits, they do not however learn about the gradients of others. **C**: The central party thus stores all gradients and hessians for the different nodes (and if secure aggregation is not used for every party as well). Together with this the splits are stored.

To find the splits, DDSketch [29] was used centrally on all data to simulate a secure federated quantile sketching algorithm.

1) Mimicking the $\mathcal{D}_{shadow}$ dataset: To perform an attack it is best to create a shadow dataset $\mathcal{D}_{shadow}$ that is as close to the target model's dataset $\mathcal{D}_{train}$ as possible. For the attacks simulated for this paper, another subset of the used dataset is used to attack the original dataset. This is done to look at the upper-bound effectiveness of the attack such that the attacker has all theoretically possible advantages.

B. Data sets

1) Healthcare: The "Healthcare Analytics II" dataset from Kaggle is a dataset on which a model can be trained to predict the length of stay for a patient. This can help hospitals by having better resource allocation. The stay duration is split between 11 classes ranging from 0-10 to more than 100 days resulting in 11 multi-class labels. There are 17 features, of which 16 features are used in testing (case_id is dropped as it does not add to the problem and could help in membership inference). The dataset has about 318k rows as only $test\_data.scv$ was used.

2) Synthetic: The synthetic dataset is created using sklearn's $make\_classification$ method was used to create a custom dataset with desired features, rows and classes. All experiments use the following parameters: $class\_sep = 1.0$, $n\_clusters\_per\_class = 1$, $n\_informative = 5$, $n\_redundant = 0$). The target classes are uniformly distributed. Different amount of multi-class labels are tested, "synthetic-10" is using 10 multi-class labels, where as "synthetic-50" is using 50 multi-class labels.

C. Experiment 1: FederBoost-Central

The Membership Inference Attack can be done on the end model. The central server can use background information by using a dataset that has the same splits as agreed upon in the target model. The attack can be done in the way described in Salem *et al.* [15] and explained in Figure 1. This attack as in Salem *et al.* will be referred to as the "Centralised" attack as no federated information is used.

1) The Centralised Attack: To see what effect the different regularisation metrics have on the effectiveness of the Membership Inference Attack, the centralised attack will be applied to these different metrics. The attack will be executed as in Salem *et al.* [15]. The attack is applied to the custom FederBoost implementation.

The configurations used for the target, shadow and attack model can be found in Table III.

D. Experiment 2: FederBoost-Federated-1

With the attack from Salem *et al.* [15] only the probability scores and thus the weights were used to do the Membership Inference Attack. As in Nasr *et al.* [28], the differentials leaked when no secure aggregation is used, could be exploited to produce a stronger attack. If secure aggregation is not used. Then the central party could attack the datasets based on the gradients of that individual participant. If secure aggregation is not used then the aggregated gradients and hessians of that single participant can also be used to get a better understanding of the individual participant's dataset. If no regularisation is used, and trees are allowed to go unreasonably deep, individual gradients could be retrieved. Even if some regularisation is used, the question remains as to how much information is in the tree split values, tree split features, and differentials. One main issue compared to the Nasr *et al.* [28] approach is that there are different amounts of previous taken splits for the nodes at different levels. Neural Networks have a static amount of parameters, and can therefore easily be put into other machine learning models. To combat this issue, an attack model is created for every depth. This attack model is tasked with only looking at nodes at that level. To combine the outputs of these different models an overarching machine learning model can be used to make a final judgement.

Parameter	target & shadow	attack model
model	FederBoost	XGBClassifier
alpha	0.0	0.0
gamma	0.5	0.5
lambda	0.1	0.1
learning rate	0.3	0.3
max_depth	8	12
$\|T\|$	20	20
nBuckets	100	-
train size	10.000	20.000
objective	softmax	binary:logistic
tree_method	hist	exact

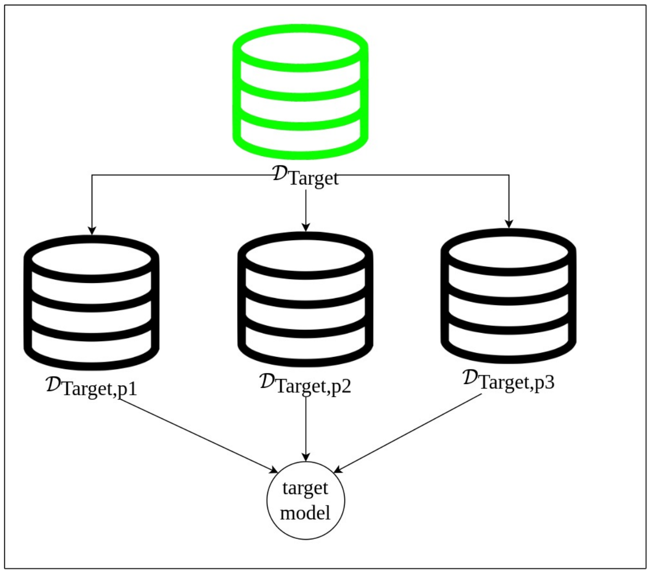
TABLE III: Configurations for target, shadow and attack model used in experiment 1.

Using the information that is leaked from the shared quantiles, the attacker could create a dataset that looks similar to the dataset that is being used by the participating party. Figure 4 further illustrates the information leakage that is captured during the federated learning process. The attack that uses the differentials, split values, and split features will be called **"FederBoost-Federated-1"**.

1) Architecture in more detail: The central party acquires most information as other parties only acquire the end model. During the simulations done for experiment 2, the to be tested dataset is split up between participants; and is labeled as in Figure 5 using Algorithm 2 and Equation 12. The "labeler" replaces the label of a data entry with 1 if it was used in training and a 0 if not. These datasets are then used to train the target model using the federated process of FederBoost.

$$label(x) = \begin{cases} 1 & \text{if } x \in \mathcal{D}_{in} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

The attack itself works quite similar to the "normal" Membership Inference Attack. Multiple shadow models are build to acquire multiple captures of information leakage of the federated process. The gradients, hessians, and splits are then used together with the class probabilities returned by the shadow model.



$\mathcal{D}_{\text{Target}}$
$\mathcal{D}_{\text{Target,p1}}$
$\mathcal{D}_{\text{Target,p2}}$
$\mathcal{D}_{\text{Target,p3}}$
target model

Fig. 5: The main database $\mathcal{D}$ that is being tested is split up between the users in the tested simulations with these datasets the main database is trained.

Algorithm 5 Centralised Membership Inference

```
Input: $\mathcal{D}_{shadow,Out}$, $\mathcal{D}_{shadow,In}$, $\mathcal{D}_{target}$ & $\mathcal{D}_{test}$
 1: $target\_model.fit(\mathcal{D}_{target})$
 2: $shadow\_model.fit(\mathcal{D}_{shadow})$
 3: $\mathcal{D}_{attack1} = labeler(\mathcal{D}_{shadow,out}, \mathcal{D}_{shadow,In})$
 4: $\mathcal{D}_{attack1}.X = shadow\_model.predict\_proba(\mathcal{D}_{attack1}.X)$

 5: $attack\_model.fit(\mathcal{D}_{attack1})$
 6: $\mathcal{D}_{shadow,In} = (X_{shadow,In}, y_{shadow,In})$
 7: $shadow\_model.fit(\mathcal{D}_{shadow,In})$
 8: $X_{test\_attack} = D_{test}.X \cup D_{target}.X$    # take only X array
 9: $y_{test\_attack} = label(\mathcal{X}_{test\_attack})$
10: $\mathcal{D}_{test\_attack} = (X_{test\_attack}, y_{test\_attack})$
11: retrieve metrics on $\mathcal{D}_{test\_attack}$
```

In this experiment the extra information that gets leaked during training is used by the server to create a Membership Inference Attack that uses the differentials. For this experiment the data holding participants are the target of the attack. Both individually as collectively the question is asked whether a dataset was in the individual participant's dataset or in the collective's dataset. A multi-class problem dataset D is split up into D_{target} and D_{server} and D_{test}. D_{target} and D_{test} are of size $train\_size$. D_{target} is split up evenly between the participating data holders in the federated network to train collaboratively. D_{test} is later used to asses how well the attack worked. Dataset D_{Server} is of size $(train\_size/2) * n$ where n is the amount of shadow models. The attacking server spits up D_{server} into $\{\mathcal{D}_{shadow,0}, \mathcal{D}_{shadow,a}, ..., \mathcal{D}_{shadow,n}\}$ each one of the size as $\frac{train\_size}{2}$. n shadow models are trained in the exact same fashion as the target model which uses $\mathcal{D}_{target}$. The shadow models are, however, trained by using $\mathcal{D}_{shadow,a}$ and $\mathcal{D}_{shadow,(a+1)\%n}$ on $shadow_a$. At this point different attacks are possible that use the available data in different manners. By going through the different shadow models with a modulo operator $\%$, it makes it such that a data entry from $\mathcal{D}_{server}$ is used both in the "In" (train) set as in the "Out" set.

The **"FederBoost-Federated-1"** attack is illustrated in Figure 6. **A**: The original dataset $\mathcal{D}$ is split up into $\mathcal{D}_{server}$, $\mathcal{D}_{target}$ & $\mathcal{D}_{test}$. $\mathcal{D}_{sever}$ is further split up into $\mathcal{D}_{shadow,0}, .., \mathcal{D}_{shadow,n}$. Such that there is a database for all n shadow models. **B**: $\mathcal{D}_{shadow,a}$ and $\mathcal{D}_{shadow,(a+1)\%n}$ are both used to train shadow model a. The union of these datasets will have the same size as $\mathcal{D}_{target}$. **C**: In the next phase, the gradients, hessians and tree structure are attempted to be retrieved. $\mathcal{D}_{shadow,a}$ and $\mathcal{D}_{shadow,(a+2)\%n}$ are used to train the attack1 models. This is done by taking differentials and split information required to get to a node on depth d and for class c as described in Equation 13. Here a dataset is created for a certain depth and class, together with the label if it was used in training. Using $label()$ as described previously in Algorithm 2 the attack1 models can thus be trained in a supervised manner.

$$\begin{aligned} X_{attack1\_c,d}(x, node) = \{x, split\_value_{d=0}, \\ splitfeatureID_{d=0}, direction_{d=0}, ..., \\ split\_value_{d}, splitfeatureID_{d}, \\ direction_{d}, node.G, node.H\} \end{aligned} \tag{13}$$

D: The attack2 model which uses the attack1 model to infer if a data entry was used during training can be trained by using $\mathcal{D}_{shadow,(a+1)\%n}$ and $\mathcal{D}_{shadow,(a+3)\%n}$. These are again labeled using Algorithm 2 to tell the attack2 model if an entry was used in training. The dataset to train attack2 is made by inferring the different attack models with nodes of the shadow model that are found on depth d and for class c. The binary probability outputs of the attack1 models are saved for every node as in Equation 14. For every depth and class the average, minimum and maximum probability returned is given to the attack2 model to train as done in Equation 15.

$$\begin{aligned} z_{c,d} = p(attack1_{c,d}(X_{attack1\_c,d} \\ (node \mid node\ from\ T\ where\ c,\ d\ and\ node\ is\ a\ leaf))) \end{aligned} \tag{14}$$

$$X_{attack\_2}(node) = \{x, model(x), avg(z_{0,0}), ..., avg(z_{c,d})\} \tag{15}$$

E. Experiment 3: FederBoost-Federated-2

The Membership Inference Attack can be done more easily by only using the gradients and hessians to calculate personal leaf node weights for every participant. When secure aggregation is not used, the main server can collect the gradients and hessians for every participant, this can then be used to calculate participant bound weights. These weights should reveal more about the subset of the participant. The attack from Salem

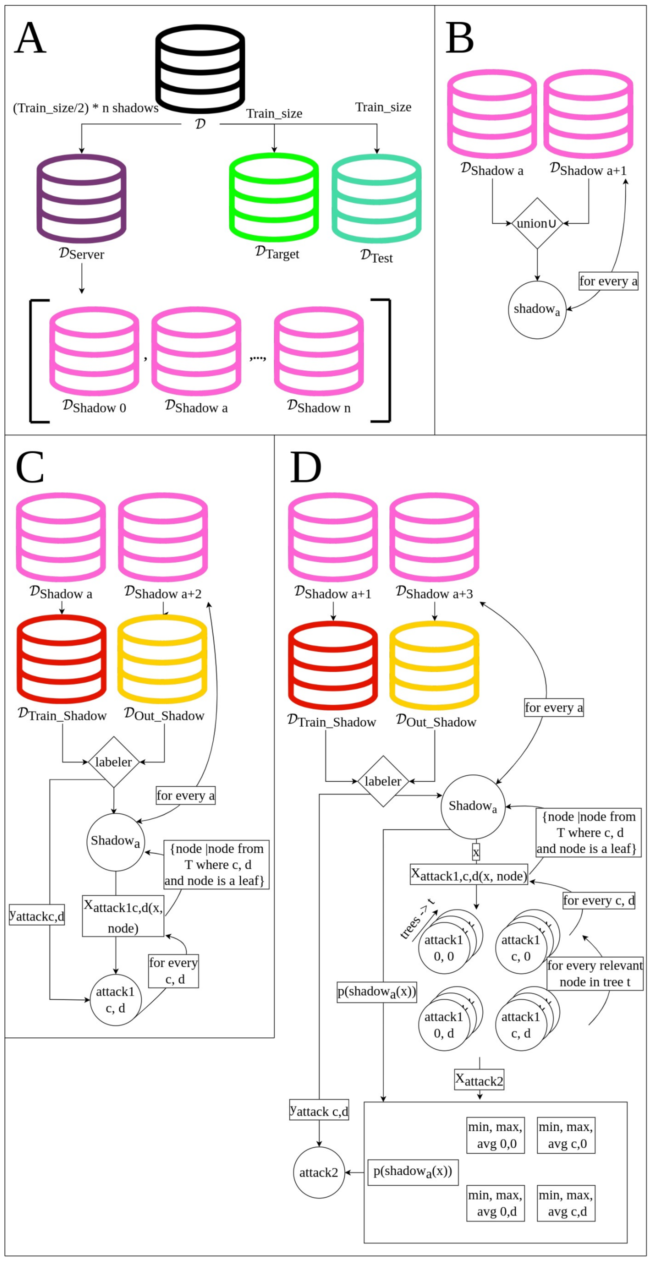

A
(Train_size/2) * n shadows
$\mathcal{D}$
Train_size
Train_size
$\mathcal{D}_{Server}$
$\mathcal{D}_{Target}$
$\mathcal{D}_{Test}$
$\mathcal{D}_{Shadow\ 0}$
$\mathcal{D}_{Shadow\ a}$
$\mathcal{D}_{Shadow\ n}$
B
$\mathcal{D}_{Shadow\ a}$
$\mathcal{D}_{Shadow\ a+1}$
union∪
for every a
$shadow_a$
C
$\mathcal{D}_{Shadow\ a}$
$\mathcal{D}_{Shadow\ a+2}$
$\mathcal{D}_{Train\_Shadow}$
$\mathcal{D}_{Out\_Shadow}$
labeler
for every a
$Shadow_a$
{node |node from T where c, d and node is a leaf}
$y_{attackc,d}$
$X_{attack1c,d}(x, node)$
for every c, d
attack1 c, d
D
$\mathcal{D}_{Shadow\ a+1}$
$\mathcal{D}_{Shadow\ a+3}$
$\mathcal{D}_{Train\_Shadow}$
$\mathcal{D}_{Out\_Shadow}$
for every a
labeler
$Shadow_a$
{node |node from T where c, d and node is a leaf}
x
$X_{attack1,c,d}(x, node)$
for every c, d
trees -> t
attack1 0, 0
attack1 c, 0
for every relevant node in tree t
$p(shadow_a(x))$
attack1 0, d
attack1 c, d
$X_{attack2}$
$y_{attack\ c,d}$
min, max, avg 0,0
min, max, avg c,0
attack2
$p(shadow_a(x))$
min, max, avg 0,d
min, max, avg c,d

Fig. 6: The FL enhanced attack "FederBoost-Federated-1" used in experiment 2.

et al. [15] can be adapted to use these participant bound weights instead of the aggregated ones. Thus the G_{Pi} and H_{Pi} as also illustrated in Figure 4. This information is used by Equation 6 to end up with a weight specific to a participant. This attack is called the **"FederBoost-Federated-2"** attack. It is compared to XGBoost and FederBoost (**"FederBoost-Centralised"**) under the Salem *et al.* [15] attack. Two different scenarios are taken into consideration, a target model with little to no regularisation and a regularised target model. The confirguration of both models can be found in Table IV. The configurations for the attack model can also be found in Table IV.

Parameter	little regularised	regularised	attack model
model	FederBoost	FederBoost	XGBClassifier
objective	softmax	softmax	binary:logistic
max_depth	10	10	12
tree_method	hist	hist	exact
learning rate	0.3	0.3	0.3
$\|T\|$	20	20	20
gamma	0	1	0
alpha	0.0	1	0
lambda	0	10	1
data division	[0.1, 0.9]	[0.1, 0.9]	-
nBuckets	100	100	-

TABLE IV: The configurations for models in experiment 3

V. Results

A. Metrics

1) Accuracy: The attack model can give one binary answer, either the given dataset was used or it was not. Accuracy is the amount of correctly classified samples divided by the amount of wrongly classified samples. For the attack the testing dataset is split 50/50 between entries used in training and entries not used in training. Thus here an an accuracy of 0.5 is as good as randomly guessing. For the different datasets the accuracy can differ for the different data problems. This metric will mostly be used to see how well the attack model can differentiate between entries used and not used in training.

2) Precision: The precision of the attack can be measured by taking the True Positive and divide it by the False Positive plus the True Positives. The output can be a useful indicator: if the precision hits 1.0 like in Fredrikson *et al.* [12], it means that if the algorithm claims a given input was in the training data, then it is guaranteed that this is true. This implicates a strong privacy breach.

3) Recall: The recall or sensitivity is used on the target model to find out how well the model is able to find all entries used in training.

3) "Training-Test Accuracy Delta" ($T-TAcc\Delta$) To measure how well regularisation is working the accuracy between the test and train dataset can be taken as used in Liu *et al.* [30]. Thus $T - TAccc\Delta = accuracy_{train}$ - $accuracy_{test}$, as the training accuracy is typically higher then the test accuracy, this metric lies between 0 and 1. This difference shows how well the algorithm was able to generalize on the test set. It is thus desired to have a low level of $T - TAcc\Delta$.

B. Experiment 1

By using the attack described by Salem *et al.* [15] different regularisation parameters can be used to defend against the Membership Inference Attack. The gamma, lambda, alpha, number of trees $|T|$, number of buckets $nBuckets$, maximum depth, learning rate, and training size were tested in this experiment. As found in other papers for different machine learning models, the story that more regularisation results in less susceptibility to the Membership Inference Attack holds.

1) Conclusions: The link between using little regularisation and accuracy for the attack can be seen in the tables mentioned below. The different datasets show significant differences to changes in accuracies, this can for example be seen in Table V.

Multi-class with more target classes are are also more susceptible to the Membership Inference Attack, as can be seen when comparing the results from the synthetic-10 and synthetic-100 datasets. The number of target classes are distributed uniformly in these datasets. For a target class there are therefore 10 times less training samples in the synthetic-100 dataset.

Table V shows the influence of different maximum depth values. The deeper the tree becomes, the higher the $T-TAcc\Delta$. The accuracy of the target model on the test set does increase. with large differences in accuracy, 0.50 to 0.73 and 0.50 to 0.78; the maximum depth of a tree is one of the more influential parameters in terms of the attack. It is at the same time a double edge sword, as the accuracy of the target model also increases with a deeper tree.

Table VI shows the influence of different alpha regularisation metric. Alpha is the L1 regularisation parameter which can range from 0 to infinity. The higher alpha is the more conservative the model is. This can be backed up with the findings, as the alpha increases the $T-TAcc\Delta$ decreases and the effectiveness of the attack decreases. Interesting is that the accuracy of the target model can perform nearly as well or even better with a larger alpha, while decreasing susceptibility to the attack. The at first highly confident synthetic-100 target model with an $T-TAcc\Delta$ of 0.77 decreases to 0.23 just by changing the alpha. While "only¨ losing 0.03 in accuracy.

Table VII shows the influence of the gamma regularisation metric. Gamma signifies how large the loss should be before a split can be taken. A larger gamma results in more pruning. The $T-TAcc\Delta$ can be seen going down with larger gamma values. The attack accuracy also decreases with a larger gamma. The accuracy does not have to suffer for this trade-off, on the contrary, the accuracy increases greatly for both the healthcare and the synthetic-10 dataset; going from 0.11 to 0.39 and 0.31 to 0.75 respectively.

Table VIII shows the influence of changing the number of estimators created. More estimators can greatly help in the accuracy of a target model, this however, comes at the cost of creating a less regularised model. Interesting is that the healthcare dataset does not get influenced significantly by this metric in terms of accuracy to the attack with only an insignificant 0.05 increase between 5 and 150 trees.

Table IX shows the influence of changing the learning rate on the attack. Larger learning rates correspond to lower levels of regularisation.

Table X shows the influence of the regularisation metric lambda. A larger lambda results in smaller loss scores, and thus a more conservative model as it acts as the L2 regularisation metric. The impact of a well-chosen lambda value is most apparent in the synthetic-100 dataset with the weakest regularization. The model becomes more useful with the highest test accuracy scores for the target model, while achieving a low accuracy/recall pair on the attack.

Table XI shows the influence of different training sizes to the attack. In general, a smaller dataset leads to a less well-regularized model for Gradient Boosted Decision Trees for the tested datasets which is more susceptible to the attack. A larger training size also results in better accuracies in the synthetic-10 and synthetic-100 datasets.

Table XII shows the influence of the number of buckets chosen for quantile sketching. The more buckets the worse the regularisation is. The effect for these datasets is, however, minimal. This could differ greatly for different untested datasets. Only a 0.11 increase in the attack accuracy for the most easily to potentially overfit synthetic-100 dataset, shows this is not always an important parameter.

C. Experiment 2

FederBoost was tested by using the attack described in Section IV-D against different datasets. The algorithm was not able to achieve meaning full accuracies across the testing. The testing dataset consists of 50% samples that were used in training and 50% samples that were not used in training.

The federated attack was tested by using both the differentials of an individual, and with the aggregated differentials G, H. When attacking the aggregated differentials a dataset size of 2000 was used, where as when attacking a single individual a data size of 25 for that individual and 2000 for total size of all participants aggregated.

1) Conclusions: Both when attacking one user with a limited database size (200) or when attacking both users using the differentials (2000), no meaningful gain in accuracy, precision or AUC was made. Apart from the stated experiment other parameters were attempted with different attack. In the time allotted, none of these attacks succeeded during this research. The full discussion about the negative results can be found in Section VII.

max_depth	healthcare				synthetic-10				synthetic-100			
	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target
5	0.50	0.43	0.11	0.01	0.50	0.34	0.10	-0.00	0.50	0.54	0.12	0.25
8	0.51	0.41	0.20	0.19	0.55	0.52	0.71	0.21	0.70	0.72	0.21	0.77
12	0.67	0.66	0.34	0.60	0.65	0.71	0.80	0.20	0.78	0.86	0.22	0.78
15	0.73	0.75	0.36	0.62	0.66	0.72	0.81	0.19	0.78	0.85	0.22	0.78

TABLE V: FederBoost-Central's attack metrics on max_depth. With the configurations found in Table III.

alpha	healthcare				synthetic-10				synthetic-100			
	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target
0	0.51	0.45	0.22	0.19	0.55	0.52	0.70	0.21	0.70	0.71	0.21	0.77
0.1	0.50	0.53	0.19	0.17	0.54	0.48	0.70	0.21	0.69	0.70	0.21	0.76
0.25	0.51	0.49	0.23	0.21	0.54	0.51	0.71	0.21	0.68	0.71	0.21	0.76
0.5	0.51	0.48	0.22	0.18	0.54	0.50	0.72	0.21	0.66	0.67	0.21	0.73
0.75	0.51	0.53	0.23	0.15	0.54	0.51	0.71	0.20	0.64	0.67	0.20	0.73
1	0.50	0.49	0.22	0.15	0.53	0.49	0.72	0.19	0.63	0.66	0.20	0.71
10	0.50	0.54	0.22	0.03	0.51	0.52	0.72	0.12	0.50	0.50	0.18	0.23

TABLE VI: FederBoost-Central's attack metrics on alpha. With the configurations found in Table III.

gamma	healthcare				synthetic-10				synthetic-100			
	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target
0	0.51	0.43	0.11	0.14	0.52	0.43	0.31	0.14	0.69	0.67	0.20	0.77
0.1	0.51	0.35	0.11	0.13	0.52	0.44	0.39	0.17	0.70	0.70	0.20	0.76
0.25	0.52	0.27	0.13	0.15	0.53	0.53	0.65	0.22	0.71	0.73	0.20	0.77
0.5	0.50	0.48	0.20	0.18	0.55	0.52	0.70	0.22	0.70	0.72	0.21	0.76
0.75	0.51	0.48	0.26	0.23	0.55	0.56	0.74	0.19	0.69	0.72	0.21	0.76
1	0.51	0.47	0.31	0.23	0.55	0.57	0.76	0.18	0.66	0.71	0.20	0.76
5	0.50	0.61	0.37	0.07	0.51	0.51	0.77	0.11	0.52	0.54	0.18	0.38
10	0.51	0.51	0.39	0.05	0.52	0.50	0.75	0.08	0.50	0.49	0.15	0.20

TABLE VII: FederBoost-Central's attack metrics on gamma. With the configurations found in Table III.

$\|T\|$	healthcare				synthetic-10				synthetic-100			
	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T - TAcc\Delta$ target
5	0.50	0.50	0.17	0.09	0.50	0.48	0.52	0.14	0.53	0.56	0.17	0.45
10	0.50	0.49	0.17	0.14	0.52	0.47	0.61	0.20	0.59	0.60	0.18	0.66
20	0.51	0.49	0.22	0.19	0.55	0.49	0.69	0.21	0.70	0.71	0.21	0.77
30	0.52	0.44	0.20	0.23	0.57	0.54	0.73	0.22	0.75	0.80	0.22	0.77
50	0.53	0.49	0.23	0.26	0.60	0.59	0.78	0.20	0.76	0.84	0.22	0.78
100	0.54	0.39	0.25	0.31	0.62	0.67	0.81	0.18	0.77	0.83	0.22	0.78
150	0.55	0.38	0.27	0.35	0.63	0.66	0.82	0.18	0.76	0.82	0.22	0.78

TABLE VIII: FederBoost-Central's attack metrics on $|T|$. With the configurations found in Table III.

learning rate	healthcare				synthetic-10				synthetic-100			
	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target
0.1	0.50	0.43	0.18	0.11	0.51	0.48	0.58	0.18	0.60	0.60	0.20	0.63
0.25	0.51	0.40	0.21	0.16	0.55	0.49	0.70	0.20	0.68	0.68	0.20	0.74
0.5	0.52	0.50	0.20	0.25	0.57	0.55	0.75	0.22	0.68	0.74	0.19	0.81
0.75	0.53	0.53	0.26	0.29	0.59	0.61	0.77	0.21	0.61	0.64	0.15	0.85
1	0.53	0.45	0.24	0.34	0.61	0.67	0.79	0.21	0.50	1.00	0.02	0.00

TABLE IX: FederBoost-Central's attack metrics on learning rate. With the configurations found in TableIII.

lambda	healthcare				synthetic-10				synthetic-100			
	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target
0	0.52	0.46	0.22	0.19	0.54	0.52	0.70	0.21	0.70	0.74	0.20	0.78
0.1	0.51	0.48	0.21	0.19	0.55	0.55	0.72	0.22	0.70	0.71	0.20	0.77
0.25	0.51	0.46	0.22	0.19	0.55	0.51	0.71	0.21	0.69	0.71	0.22	0.75
0.5	0.51	0.44	0.24	0.18	0.54	0.50	0.73	0.20	0.68	0.67	0.22	0.72
0.75	0.51	0.52	0.25	0.19	0.55	0.56	0.74	0.20	0.67	0.63	0.22	0.70
1	0.51	0.52	0.23	0.18	0.54	0.56	0.73	0.19	0.67	0.66	0.23	0.69
10	0.50	0.55	0.37	0.11	0.53	0.55	0.76	0.13	0.56	0.57	0.24	0.43

TABLE X: FederBoost-Central's attack metrics on lambda. With the configurations found in Table III.

train_size	healthcare				synthetic-10				synthetic-100			
	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target
1000	0.67	0.71	0.31	0.64	0.72	0.76	0.58	0.42	0.85	0.90	0.07	0.93
2000	0.59	0.66	0.31	0.60	0.69	0.70	0.61	0.39	0.84	0.89	0.12	0.88
5000	0.54	0.54	0.28	0.34	0.60	0.57	0.69	0.28	0.77	0.80	0.19	0.80
10000	0.51	0.43	0.23	0.19	0.54	0.52	0.72	0.20	0.70	0.72	0.20	0.77

TABLE XI: FederBoost-Central's attack metrics on train_size. With the configurations found in Table III.

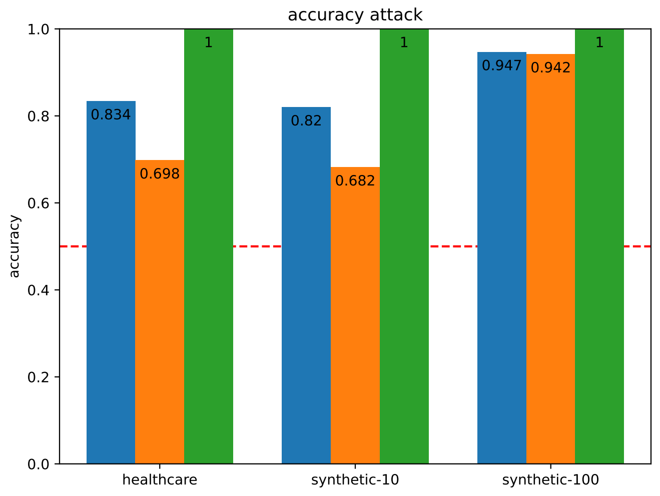
nBuckets	healthcare				synthetic-10				synthetic-100			
	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target	accuracy attack	recall attack	accuracy target	$T-TAcc\Delta$ target
10	0.50	0.51	0.21	0.17	0.53	0.53	0.72	0.17	0.62	0.62	0.20	0.72
20	0.50	0.48	0.21	0.18	0.53	0.55	0.73	0.19	0.66	0.67	0.21	0.75
50	0.51	0.44	0.22	0.19	0.54	0.53	0.73	0.20	0.69	0.71	0.21	0.76
100	0.51	0.42	0.23	0.19	0.55	0.52	0.70	0.21	0.70	0.74	0.21	0.77
1000	0.51	0.44	0.21	0.19	0.55	0.54	0.69	0.23	0.71	0.73	0.20	0.78

TABLE XII: FederBoost-Central's attack metrics on nBuckets. With the configurations found in Table III.

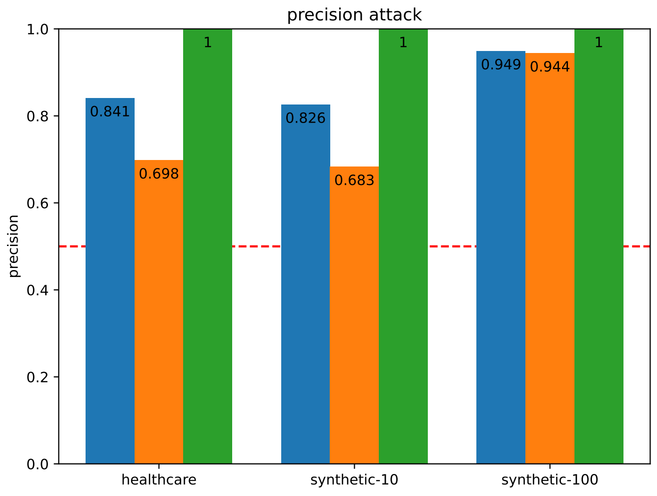
D. Experiment 3

The FederBoost-Federated-2 attack was able to use the send differentials to perform the Membership Inference Attack with higher accuracies than without using the leaked federated information. Figure 7 shows that an individual participant of a poorly regularised federated learning setup can be attacked more effectively when using federated information. The accuracy of the **"FederBoost-Federated-2"** attack achieving accuracies and precision values of 1. It is thus able to classify all 200 of participant's P_1's data entries, and separate the other 200 data entries that were not used in training. Figure 8 shows the loss over time for the different datasets for the training and test dataset. This shows that the healthcare dataset is overfitting. The synthetic-10 dataset is not overfitted and is still improving, and the synthetic-100 dataset is slightly overfitted, here.

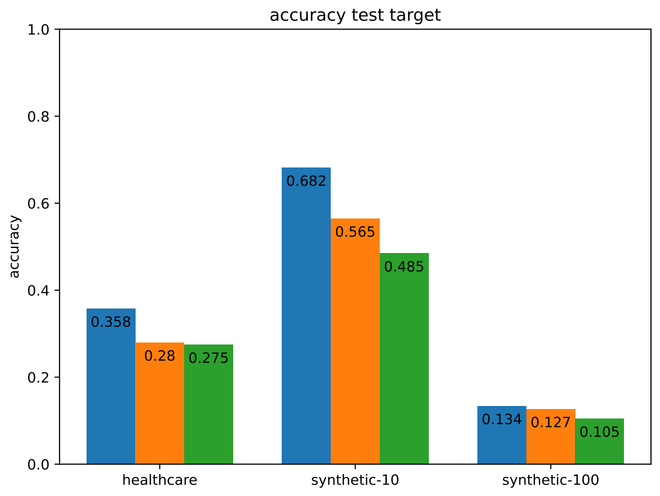
However as seen in Figure 9, when the target model was regularised well using the different regularisation metrics, the attack is not able to make significant gains. The attack's accuracy on the global weights is about the same as on the weights of the first participant P_1.



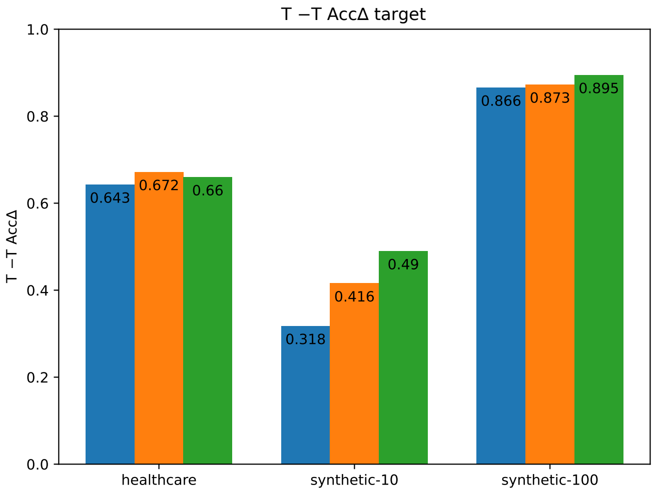
(a) Accuracy of the attack



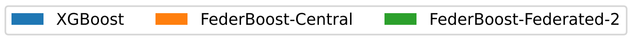
(b) Precision of the attack



(c) Accuracy of the target



(d) $T - TAcc\Delta$ of the target



(e) Legend

Fig. 7: Experiment 3, using an target model made with little regularisation.

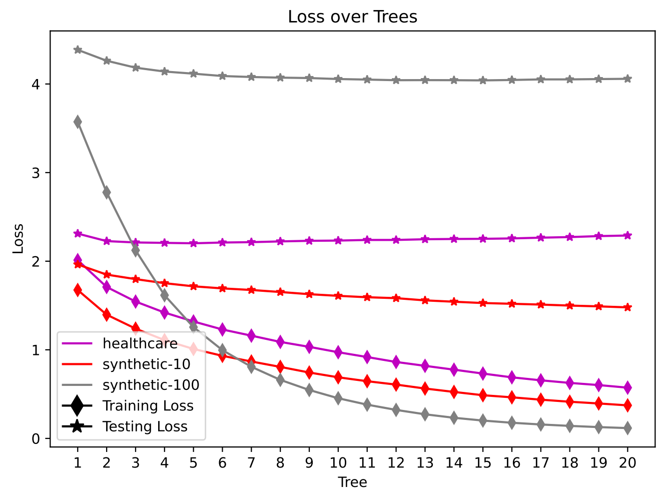


Fig. 8: Loss over time for FederBoost's low regularisation scenario

VI. Related Work

This section will discuss previous papers that discuss privacy breaching attacks on machine learning, different defences against these attacks, and different federated algorithms will be discussed. An overview of the different papers discussed in the sections about the attacks and defences can be found in Table XIII in Appendix A. Here for the trust scenario, a user can be "honest-but-curious", this means that the party will follow the protocol. An malicious attacker does not have to follow the protocol.

A. Membership Inference

1) Attack: Apart from Shokri *et al.* [5] and Salem *et al.* [15] research into Membership Inference Attacks as described in Section II-D, there are other papers that look into attacking other models like GBDT's or DT's.

Yuan *et al.* [31] proposes a self-attention Membership Inference Attack, this attack uses different thresholds between the confidence gaps to provide a more "fine-grained Membership Inference Attack. This works by inputting the confidence, sensitivity and one-hot label into the attack model which uses multi-head self-attention modules which aims to capture global dependencies among inputs and allows the inputs to interact with each other.

Liu *et al.* [30] examined attacking the logistic regression, XGBoost and cloud machine learning algorithms using the Membership Inference Attack. They introduce SocInf, a construction for conducting Membership Inference Attacks using a GAN to mimic any type of model. The models are inferenced upon in a black-box fashion. The authors found a clear correlation between the level of overfitting and effectiveness of the attack. The attack on XGBoost on different datasets achieved an average accuracy of 0.7339 on their used datasets.

de Arcaute *et al.* [20] assessed the impact of Membership Inference for different classical machine learning approaches. In their experiments they observed that a Membership Inference Attack was less effective when used against classical machine learning models. The authors suggested that this might be due to the lower risk of overfitting. However this conclusion is subject to debate.

Truex *et al.* [14] also investigated the effectiveness of the Membership Inference Attack on different classical machine learning approaches. In this paper however Decision Trees were the worst against Membership Inference Attacks. The highest accuracy of the Membership Inference Attack here was 95.74 in strong contrast to de Arcaute *et al.* who concluded that the attack did not seem effective on Decision Trees. Truex *et al.* argues the following about Membership Inference Attacks:"(1) they are data-driven attacks, (2) attack models are transferable, (3) target model type is a strong indicator of model vulnerability, (4) attack data generation techniques need not explicitly mirror the target model, and (5) Membership Inference Attacks can persist as insider attacks in federated systems." Truex *et al.* applied the attack from Shokri *et al.* [5].

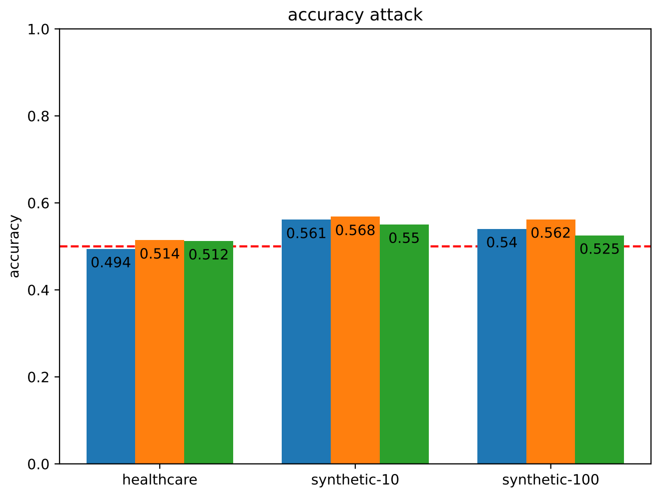
Nasr *et al.* [28] used one custom created Neural Network to attack the model in a similar fashion to Salem *et al.* [15]. Their highest attack accuracy reported was 0.676.

Nasr, Shokri & Houmansadr [18] conducted a comprehensive study on white-box adversarial attacks on deep Neural Networks using Membership Inference. They implement two different attacks, one using background knowledge to create a supervised attack model, and one that uses an unsupervised attack model. For the unsupervised attack an auto-encoder is created that is made to do Membership Inference on any data. They claim that "even well generalized deep models might leak significant amount of information about their training data, and could be vulnerable to white-box Membership Inference Attacks". This attack is also made easier for the local participants and the central server in a Federated Learning setting.

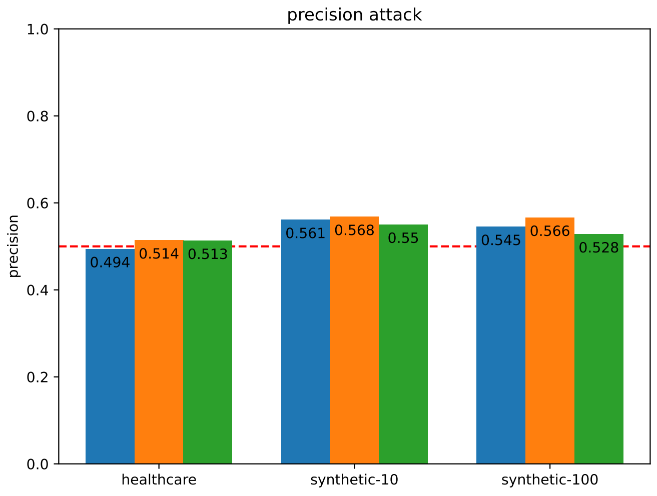
Melis [32] used model updates during the federated learning process to successfully conduct a Membership Inference Attack. This attack was used on a federated setup where the central server would be updated for each mini-batch instead of each epoch. This made it such that the central server can save the differences of weights created by one mini-batch. By holding this information on only a mini-batch, the attacker receives a lot of information about the original data entry.

2) Defence: Regularization can be applied to any machine learning approach. Truex *et al.* [14] explains that Model Hardening (model choice, fit control, regularisation, anonymization), API hardening and Differential Privacy can be used to protect against Membership Inference Attacks as well.

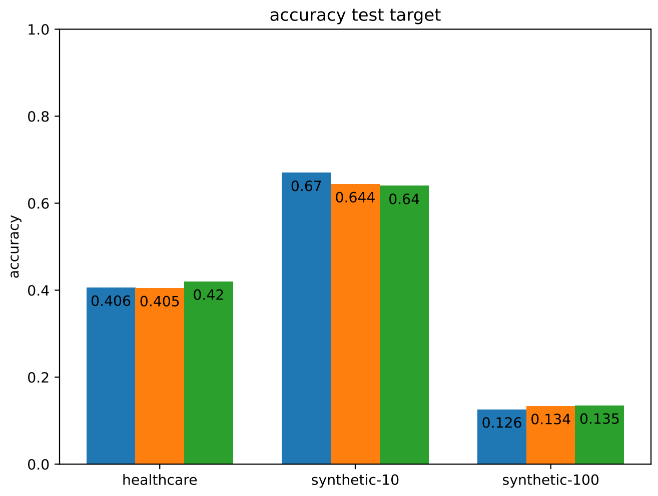
In a paper from Jia *et al.* [33] a Membership Inference defence was proposed called MemGuard. MemGuard carefully crafts the confidence scores such that Membership Inference (supposedly) cannot be done on the black-box model. MemGuard does not influence the target models' prediction accuracy. However, the security claims of MemGuard have



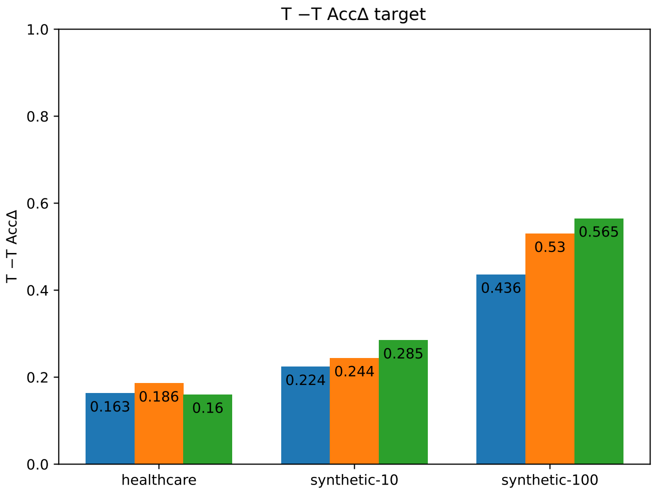
(a) Accuracy of the attack



(b) Precision of the attack



(c) Accuracy of the target



(d) $T - T\,Acc\Delta$ of the target

🟦 XGBoost 🟧 FederBoost-Central 🟩 FederBoost-Federated-2

(e) Legend

Fig. 9: Experiment 3, using an well regularised target model.

been put into question as Song and Mittal [22] were able to attack the model with higher accuracies then reported. This was possible because "MemGuard lacks consideration of strategic adversaries". This means that an attacker can use the fact that MemGuard is being used to amplify his attack.

Nasr *et al.* [28] uses a min-max game that minimises the prediction loss of the model while creating a model that can mitigate the risks of Membership Inference Attacks (near random guesses) on a black-box model. Song and Mittal [22], however, argue that adversarial regularisation like done by Nasr *et al.* is no better than early stopping in Neural Networks. This attack requires the defender to have access to the data, this is therefore not trivial to implement in a Federated Learning setting.

Next to a new attack, Yuan *et al.* [31] also proposes a new defence which uses KL-divergence distances. Their attack named "Pair-based Posterior Balancing" aims at reducing the divergence between scores of members and non-members.

B. *Federated Algorithms*

This section will show different Federated Learning implementations of XGBoost. For an overview of the different privacy-preserving algorithms discussed below the reader can refer to Table XIV in Appendix A.

1) Horizontal XGBoost: Ong *et al.* [34] proposes Party-Adaptive XGBoost (PAX). A Federated Learning setting which uses XGBoost without encryption. Without encryption

PAX is significantly faster then SecureBoost. As a security measure PAX makes use of the security provided by the *quantile-based approximation* that XGBoost provides. By putting the original data in bins individual user's features are combined and obfuscated. The "Party-Adaptive" part of PAX lets different users calculate their privacy parameter ϵ in an effort to make sure deviations in other party's data do not effect the learning. In contrast to most other algorithms, only one histogram is build in the beginning of the algorithm.

A more recent paper in which Jones *et al.* [35] build a simplified version of Ong *et al.* 's implementation. In this implementation the "Party-Adaptive" part of the algorithm is not being used, parties create histograms with the same bin size such that they can more easily be merged.

Cheng *et al.* [36] has a version SecureBoost that supports the horizontal setting. In this version the passive nodes will create their histograms of gradients and hessians and add random numbers to it such that the random numbers cancel each other out. This is done by using secure aggregation using Additive Homomorphic Encryption. The main server therefore only learns the aggregated gradients and hessians. The central server can then find the best split and communicate it with the other participants. After a split is received the passive nodes will create a new histogram with different samples from the dataset. This is continued until max depth is reached or other stop conditions are reached. The process will repeat until the desired amount of trees are constructed. This process can be quite slow as pointed out in Ong *et al.* [34]. This is because the amount of communication that has to happen together with the expensive homomorphic encryption.

Tian *et al.* [23] proposes FederBoost for both vertical as Horizontal Federated Learning. The GBDT algorithm almost preforms as well as a central XGBoost implementation in their findings. For the horizontal setting only lightweight secure aggregation is needed. Quantiles are generated at the start the algorithm such that they can be used for every tree. New gradients are calculated for every split, these gradients are aggregated into the respective quantile bins. The different gradients and hessians are send to the central party using secure aggregation. This prevents that the central party learns about about the gradients and hessians of a single party.

Yang *et al.* [37] proposes a Federated XGBoost which is also available on GitHub[1]. Federated XGBoost uses data aggregation and k-anonymity to achieve their level of desired privacy. The algorithm further works in much of the same fashion as FederBoost, however without the secure aggregation.

Liu *et al.* [38] proposes FedXGB, designed against user dropout and forced aggregation. Forced aggregation means that the server is forced to aggregate the encrypted model updates from the participants. This privacy-preserving algorithm uses both homomorphic encryption as secret sharing to create a secure training environment. This algorithm scales linear with both the amount of users as the input size.

[1]https://github.com/Raymw/Federated-XGBoost

2) Horizontal GBDT: Wu *et al.* [39] introduces Pivot-DT, Pivot-RF and Pivot-GBDT. This paper presents a way to do Federated Decision Tree learning without a trusted third party. And provides protection against parties that are listening into the communication. The model stays encrypted and all users need to be online to get a prediction.

C. Property Inference

The Property Inference Attack is another attack that aims to infer information about a dataset. The Property Inference Attack aims to infer overall properties of the training data. Ganju *et al.* [6] examined white-box Fully Connected Neural Networks (FCNN) to retrieve information about the training dataset that is supposed to be private. The authors give the example of inferring *global* properties of a malware detection model. The properties of the testing environment that effect *all* of the traces could be retrieved to evade detection. This can again be achieved by using "shadow" models" like in the Membership Inference Attack. These models are trained on datasets, half of the datasets have a property P the other half $\bar{P}$ does not have this property. The features (weights and biases) are retrieved and normalised by sorting weights by magnitude of sums or by using unordered sets to represent the network. A meta-classifier is then trained on these features to create a model that can detect if a model was trained with a property P.

D. Model Inversion

1) Attack: A Model Inversion Attack is an attack where parts of the training data is retrieved from the model. In Neural Network models this can be done by looking at the gradient updates, using an encoder decoder setup where the encoder is the victim's model, with a GAN, or by combining these approaches. A Model Inversion Attack can be seen as the most powerful attack as training data can be (partially) reconstructed if successful.

In Yang *et al.* [7] a Neural Network Inversion was used to create a new adversarial model. This model could be used to infer information about the training set. This inverted model can be created using using background knowledge or by training the inversion model with truncated output data from the classifier. The inverted model is the decoder in an encoder decoder network, where the encoder is the victim's model. This paper shows that Model Inversion Attacks can work in reconstructing training data by creating identifiable faces.

Fredrikson *et al.* [12] successfully attacked a Decision Tree API by using a Model Inversion Attack. When Model Inversion is applied against Decision Trees, most of the dataset's sensitive features can be inferred with a precision of 1.0. Also, the authors argued that releasing a model in a white-box manner can greatly enhance the adversary's advantage.

Hitaj *et al.* [13] concluded that "distributed, federated or decentralised deep learning approach is fundamentally broken and does not protect the training sets of honest participants". By using a GAN the training of the model can be worsened by training the model with generated data. This makes it so that the participants have to insert more of their private data into the model to achieve required accuracy levels. This attack works as long as the victim is inserting more of its label's data to train.

Wang *et al.* [17] also used a GAN to do a Model Inversion Attack. Here, however, the GAN was used by the server to spy on the white-box model. The update from the user after target data is inserted can directly be used to update the discriminator. This attack also showed the ability to successfully generate data that looks near identical to what the victim trained upon.

Zhang *et al.* [40] applied a Model Inversion Attack using a GAN, against a white-box model using auxiliary knowledge. This auxiliary knowledge can be blurred or masked images for example. Public datasets together with the white-box model's responses are used to train the GAN.

2) Defence: Friedman and Schuster [41] look at the privacy of Decision Trees, they introduce error based pruning to have differential privacy in the Decision Tree model. This paper had large variance in their experimental results. Their experiments further show a clear cost in accuracy when privacy is increased.

Fredrikson *et al.* [12] examined sensitive feature splitting at different levels in the trees, as well as reducing Neural Network API precision. Splitting the sensitive attributes at different heights did have an influence on Membership Inference and Class Accuracies. The sensitive feature splitting did not show significant improvements, rounding API returns for Neural Networks however did decrease attack accuracies.

In a paper from Park, Hong and Seo [42] different private Decision Tree models were investigated to counter Model Inversion Attacks. The authors compared three different privacy-preserving algorithms from:, Blum *et al.* [43], Friedman *et al.* [41], and Mohammed *et al.* [44]. The authors found that differentially private Decision Trees can mitigate the Model Inversion Attack. Where as the non-defended models were susceptible to Model Inversion Attacks. Friedman *et al.* [41]'s algorithm showed the best performance while also countering the Model Inversion Attack significantly.

Multi-Party Computation and homomorphic encryption can mitigate some attacks like the one mentioned in Hitaj *et al.* [13]. In cases where the model is attacked during the training phase, encryption can mitigate model information leakage.

E. Model Extraction

Tramèr *et al.* [21] provided a method to do Model Extraction on Logistic Regression, Decision Trees or Neural Networks. This Model Extraction Attack aims to rebuild the black-box model. The attack is shown to be successful in this paper thus putting black-box models in danger. Depending on the dataset the authors were able to extract an 100% equivalent model with about 1000 to 4000 queries on a German Credit and Steak Survey dataset. However, the ease of an attack like this depends on the complexity of the tree and the underlying dataset. This attack could potentially also be used against the many trees of a GBDT algorithm or an ensemble method like RF.

VII. Discussion & Future Work

Gradient Boosted Decision Trees have a large amount of parameters, most of which influential on the effectiveness of the Membership Inference Attack. The difference in the total dataset, and individual participant's datasets can all heavily influence the susceptibility to the attack. The attacks assumed that the attacker were in possession of a lot of background information; it was thus attempted to find the maximum possible effectiveness of the different attacks. The given implementation FederBoost-Federated-1 which tried to mimic Nasr *et al.* [18] was not able to produce significant enough improvements over randomly guessing in the tests. However, as FederBoost-Federated-2 shows, using the extra federated information can improve the effectiveness of the attack. A different implementation of FederBoost-Federated-1 could work, this is true because it should be able to achieve the same if not higher accuracy levels of FederBoost-Federated-2, and the centralised attack. FederBoost-Federated-1 namely uses the same, and more information as these attacks. FederBoost-Federated-1 could have failed due to an implementation mistake, wrong parameters, or wrong model choices for the attack models. This should be investigated further, to better find the maximum available effectiveness of a Membership Inference Attack that uses the extra leaked federated information. The differentials hold some information about the original dataset, extracting this information out of the many gradients and hessians, proved to be a barrier. In most testing all nodes were used to retrieve differentials from; only the leaf nodes would have to be used as the entire tree structure can be found above the leaf nodes, and the gradients and hessians of nodes above the leaf nodes are just the aggregate of two leaf nodes. Certainly starting off changing FederBoost-Federated-2 with an attack that takes the entire gradient and hessian matrix at the leaf nodes instead of the weight of a participant could already create a stronger attack. FederBoost-Federated-2 could potentially work better when different participants hold information that is different in terms of average, standard deviation, etc. An attacker might be able to catch these differences in the attack. All participants only had the same dataset distribution in the testing of this paper (although these distributions were practically different when using a really small training set).

It is also interesting to observe that XGBoost was more susceptible to the Membership Inference Attack than the custom implementation of FederBoost in the experimentation's. This difference is less significant when regularisation is introduced. As XGBoost is an open source project, the actual implementation of XGBoost is slightly different than documented. Although it was attempted to create a FederBoost that would mimic XGBoost best, XGBoost seems to have a bit less regularisation in favour of speed outside of the tested parameters.

Different defences against the federated attack would then also have to be explored further, although even some regularisation showed significant improvements against the Membership Inference Attack. Regularisation does not give a defender a guarantee, Differential Privacy [45] could provide this guarantee. Certainly, when using medical data, this absolute privacy guarantee should be pursuit. K-anonymity can also be used a privacy enhancing measure on the datasets of the participants. The use of secure aggregation is also reinforced as an important defensive measure, the privacy given by the quantile sketching can be too insignificant to reduce the effectiveness of the attack.

In the testing only a passive attacker that listens in to the communication is examined, the attack can however be exacerbated when the attacker does not follow the protocol. It could attack a participant and lie in the chosen split decisions by secretly using different regularisation parameters.

VIII. Conclusions

Gradient Boosted Decision Trees continue to show a link between the degree of using regularisation and the effectiveness of the Membership Inference Attack. If secure aggregation is not used, an attacker is able in some cases to use this extra information to create an attack with a higher accuracy. The maximum potential of this federated attack has not been reached yet, as the communicated gradients and hessians of leaf nodes were only indirectly used by looking at the participant bounded weights. The tree structure was not successfully taken into the attack as well, the combination of the gradients, hessians and tree structure could even create a stronger attack in the envisioned threat model.

A. Answering The Main Research Question

Regularisation showed to be highly important in combatting the Membership Inference Attack, thus defending the individual's privacy. Regularisation has to be used excessively or tuned correctly during all federated training to limit information leakage. At first sight the lack of secure aggregation does not seem to be a problem when using regularisation, however, the successful attack (FederBoost-Federated-2) only used weights bound to a participant to successfully create a stronger attack. The attacker could have more information at its disposal; that being the information hidden in the gradients, hessians and tree structure. The maximum effectiveness in the Membership Inference Attack in the given threat scenario is thus not fully explored and no guarantee can be given to a defender. Only secure aggregation, or differential privacy could give the defender some guarantees.

B. Answering Research Question 1

The gamma, alpha, lambda regularisation metrics all affect the split calculation, all three are able to effectively increase regularisation and in turn show the ability to limit the susceptibility to the Membership Inference Attack. The usage of both alpha and lambda in the weight calculation makes

The number of trees, and maximum depth, can both greatly influence the degree of regularisation. having a smaller training size, makes overfitting easier on Gradient Boosted Decision Trees. Using too many trees can result in extra privacy leakage. With the learning rate having a minimal impact on the degree of regularisation.

The results also show that merely relying on a low amount of buckets during quantile sketching is not enough to protect against the Membership Inference attack.

C. Answering Research Question 2

The full extend to which the extra federated learning can be exploited was not found, however, it is found that an attacker is able to use the information to it's advantage with "FederBoost-Federated-2". An individual smaller hospital could realistically have a dataset of only a couple patients. If the gradients, hessians and tree structure are all used, the attacker might be able to use the leaked federated information to mount an attack with a higher accuracy then without using the extra information. An adapted, re-implemented "FederBoost-Federated-1" could further force a defender to use secure aggregation in order to protect it's information.

Acknowledgment

I want to thank my main supervisor Florian Hahn (University of Twente), Nicola Strisciuglio (University of Twente), Albert Wong (RIVM), and Mark Kroon (RIVM) for their guidance and feedback.

References

[1] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

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

[3] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[4] W. G. Van Panhuis, P. Paul, C. Emerson, J. Grefenstette, R. Wilder, A. J. Herbst, D. Heymann, and D. S. Burke, "A systematic review of barriers to data sharing in public health," *BMC public health*, vol. 14, no. 1, pp. 1–9, 2014.

[5] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[6] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 619–633.

[7] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 225–240.

[8] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *arXiv preprint arXiv:1708.08296*, 2017.

[9] P. Angelov and E. Soares, "Towards explainable deep neural networks (xdnn)," *Neural Networks*, vol. 130, pp. 185–194, 2020.

[10] "State of data science and machine learning 2021," Oct 2021. [Online]. Available: https://www.kaggle.com/kaggle-survey-2021

[11] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *The journal of machine learning research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[12] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[13] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 603–618.

[14] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Demystifying membership inference attacks in machine learning as a service," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 2073–2089, 2019.

[15] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018.

[16] M. Xu and X. Li, "Subject property inference attack in collaborative learning," in *2020 12th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 1. IEEE, 2020, pp. 227–231.

[17] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019-IEEE conference on computer communications*. IEEE, 2019, pp. 2512–2520.

[18] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 739–753.

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

[20] G. M. R. de Arcaute, J. A. Hernández, and P. Reviriego, "Assessing the impact of membership inference attacks on classical machine learning algorithms," in *2022 18th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, 2022, pp. 1–4.

[21] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis." in *USENIX security symposium*, vol. 16, 2016, pp. 601–618.

[22] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models." in *USENIX Security Symposium*, vol. 1, no. 2, 2021, p. 4.

[23] Z. Tian, R. Zhang, X. Hou, J. Liu, and K. Ren, "Federboost: Private federated learning for gbdt," *arXiv preprint arXiv:2011.02796*, 2020.

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

[25] Y. Liu, T. Fan, T. Chen, Q. Xu, and Q. Yang, "Fate: An industrial grade platform for collaborative learning with data protection," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 10 320–10 325, 2021.

[26] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[27] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[28] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 634–646.

[29] C. Masson, J. E. Rim, and H. K. Lee, "Ddsketch: A fast and fully-mergeable quantile sketch with relative-error guarantees," *arXiv preprint arXiv:1908.10693*, 2019.

[30] G. Liu, C. Wang, K. Peng, H. Huang, Y. Li, and W. Cheng, "Socinf: Membership inference attacks on social media health data with machine learning," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 907–921, 2019.

[31] X. Yuan and L. Zhang, "Membership inference attacks and defenses in neural network pruning," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 4561–4578. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/yuan-xiaoyong

[32] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 691–706.

[33] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "Memguard: Defending against black-box membership inference attacks via adversarial examples," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 259–274.

[34] Y. J. Ong, Y. Zhou, N. Baracaldo, and H. Ludwig, "Adaptive histogram-based gradient boosted trees for federated learning," *arXiv preprint arXiv:2012.06670*, 2020.

[35] K. Jones, Y. J. Ong, Y. Zhou, and N. Baracaldo, "Federated xgboost on sample-wise non-iid data," *arXiv preprint arXiv:2209.01340*, 2022.

[36] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, and Q. Yang, "Secureboost: A lossless federated learning framework," *IEEE Intelligent Systems*, vol. 36, no. 6, pp. 87–98, 2021.

[37] M. Yang, L. Song, J. Xu, C. Li, and G. Tan, "The tradeoff between privacy and accuracy in anomaly detection using federated xgboost," 2019.

[38] Y. Liu, Z. Ma, X. Liu, S. Ma, S. Nepal, and R. Deng, "Boosting privately: Privacy-preserving federated extreme boosting for mobile crowdsensing," *arXiv preprint arXiv:1907.10218*, 2019.

[39] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," *arXiv preprint arXiv:2008.06170*, 2020.

[40] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: Generative model-inversion attacks against deep neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 253–261.

[41] A. Friedman and A. Schuster, "Data mining with differential privacy," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 493–502.

[42] C. Park, D. Hong, and C. Seo, "Evaluating differentially private decision tree model over model inversion attack," *International Journal of Information Security*, vol. 21, no. 3, pp. 1–14, 2022.

[43] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: the sulq framework," in *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2005, pp. 128–138.

[44] S. Fletcher and M. Z. Islam, "A differentially private random decision forest using reliable signal-to-noise ratios," in *AI 2015: Advances in Artificial Intelligence: 28th Australasian Joint Conference, Canberra, ACT, Australia, November 30–December 4, 2015, Proceedings 28*. Springer, 2015, pp. 192–203.

[45] C. Dwork, "Differential privacy," in *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33*. Springer, 2006, pp. 1–12.

[46] N. K. Le, Y. Liu, Q. M. Nguyen, Q. Liu, F. Liu, Q. Cai, and S. Hirche, "Fedxgboost: Privacy-preserving xgboost for federated learning," *arXiv preprint arXiv:2106.10662*, 2021.

[47] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 4642–4649.

[48] B. Liu, C. Tan, J. Wang, T. Zeng, H. Shan, H. Yao, H. Huang, P. Dai, L. Bo, and Y. Chen, "Fedlearn-algo: A flexible open-source privacy-preserving machine learning platform," *arXiv preprint arXiv:2107.04129*, 2021.

[49] J. Hou, M. Su, A. Fu, and Y. Yu, "Verifiable privacy-preserving scheme based on vertical federated random forest," *IEEE Internet of Things Journal*, vol. 9, no. 22, pp. 22 158–22 172, 2021.

[50] Y. Liu, Y. Liu, Z. Liu, Y. Liang, C. Meng, J. Zhang, and Y. Zheng, "Federated forest," *IEEE Transactions on Big Data*, vol. 8, no. 3, pp. 843–854, 2020.

[51] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Advances in Cryptology—CRYPTO 2000: 20th Annual International Cryptology Conference Santa Barbara, California, USA, August 20–24, 2000 Proceedings*. Springer, 2000, pp. 36–54.

[52] J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson, "Privacy-preserving decision trees over vertically partitioned data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 2, no. 3, pp. 1–27, 2008.

[53] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J.-P. Bossuat, J. S. Sousa, and J.-P. Hubaux, "Poseidon: Privacy-preserving federated neural network learning," *arXiv preprint arXiv:2009.00349*, 2020.

[54] S. Wagh, D. Gupta, and N. Chandran, "Securenn: 3-party secure computation for neural network training." *Proc. Priv. Enhancing Technol.*, vol. 2019, no. 3, pp. 26–49, 2019.

Appendix

A. *Related Work*

Paper	Used attack	Setting	Trust scenario	Target Network	Proposed defence
Yang, *et al.* [7]	Model Inversion	Centralised API	Black-box, honest-but-curious user & malicious creator	NN	-
Fredrikson *et al.* [12]	Model Inversion	Centralised API	White-box & black-box DT ML model API & honest-but-curious API user	DT & NN	Sensitive feature splitting & reducing NN API precision
Wang *et al.* [17]	Model Inversion	Federated	White-box & Malicious Server	NN	-
Park *et al.* [42]	Model Inversion	Centralised	Black-box, honest-but-curious user who has model	DT	Multiple differential privacy-preserving algorithms
Hitaj *et al.* [13]	Model Inversion (GAN)	Federated	Active insider in white-box training	DNN	differential privacy (useful till a certain extent)
Truex *et al.* [14]	Membership Inference	MLaaS & active FL insider	Black-Box, honest-but-curious user	DT, LR, k-NN, NB & NN	Short survey into Model Hardening, API hardening & Differential Privacy, no experimentation.
Liu *et al.* [30]	Membership Inference	Centralised API	Black-box, honest-but-curious	XGBoost, BigM & LR	Overfitting, differential privacy (no experimentation), and coarsening the precision of prediction.
Shokri *et al.* [5]	Membership Inference	Centralised API	Black-box	CNN & FCNN	Generalization
de Acraute *et al.* [20]	Membership Inference	Centralised	Black-box, honest-but-curious user	DT, RF, AdaBoost & more	-
Salem *et al.* [15]	Membership Inference	Centralised API	Black-box API & honest-but-curious API user	NN, LR & RF	Dropout & model stacking = train different parts target model trained with different subsets (ensemble learning).
Nasr *et al.* [28]	Membership Inference	Centralised	Black-box, honest-but-curious	CNN	Min-max game to defend against the attack
Nasr, Shokri & Houmansadr [18]	Membership Inference	Federated	White-box, honest-but-curious & adversarial	CNN	-
Melis *et al.* [32]	Membership Inference	Federated	White-box, honest-but-curious	CNN	-
Tramér *et al.* [21]	Model extraction attack	Centralised API	Black-box, honest-but-curious user	LR & NN & DT	-
Karan *et al.* [6]	Property Inference	Centralised	White-box, honest-but-curious user	FCNN	-
Xu *et al.* [16]	Property Inference	Federated server	White-box, honest-but-curious & malicious	CNN	-

TABLE XIII: Current literature of machine learning attacks on different machine learning approaches

Paper	Algorithm	Name	Data	Threat	Idea	Conclusion
Ong *et al.* [34]	XGBoost	PAX	H	-	FL implementation of XGBoost without encryption. The gradients, hessians and histograms will be exchanged around to create the tree.	Significantly faster than SecureBoost.
Jones *et al.* [35]	XGBoost	-	H	-	Simplify Ong *et al.* 's implementation by not doing "party-adaptive" part	The use of histograms should be private while still allow the creation of an accurate model.
Cheng *et al.* [36]	XGBoost	SecureBoost	V & H*	Honest-but-curious	Aggregate encrypted gradient statistics. FATE: GitHub implementation.	Lossless, *Although not covered in paper, it can be done with SecureBoost
Tian *et al.* [23]	XGBoost	FederBoost	V & H	Honest-but-curious	Differential Privacy for vertical setting and lightweight cryptography for horizontal setting.	Fast and comparable performance to XGBoost.
Yang *et al.* [37]	XGBoost	Federated XGBoost	H	-	Sum up gradient's of nodes and using k-anonymity. server sends out direction of split in FL setting. GitHub implementation.	
Liu *et al.* [38]	XGBoost	FedXGB	H	Honest-but-curious	XGBoost where gradient updates are shared and aggregated encrypted through a central server using cryptography.	Robust against User dropout
Le *et al.* [46]	XGBoost	FedXGBoos	V	-	Central server uses Secure Multi-Party Computation to find the best split	
Li *et al.* [47]	GBDTs	-	H	Honest-but-curious	Use Locality-Sensitive Hashing to share data and create trees together.	
Wu *et al.* [39]	DT, RF, & GBDT	Pivot	V	Honest-but-curious clients	Homomorphic encryption & MPC on training, white-box model on prediction on vertical data	Encrypted scenario performance as well as non-private scenario.
Liu *et al.* [48]	RF	Fedlearn-Algo	V	-	Provides GitHub framework with Random Forest being implemented	
Hou *et al.* [49]	RF	VPRF	V	Honest-but-curious with TTP	Homomorphic encrypted model & delegated computing verification	More efficient then Wu *et al.* 's Pivot [39]
Liu *et al.* [50]	RF	Federated Forest	V	Honest-but-curious with TTP	Using CART, master server chooses split features randomly, receives client best splits, entire model distributed among clients, master can have entire model.	Lossless classification, raw data is not exposed. Possible attacks on model not mentioned.
Friedman *et al.* [41]	DT	DiffPID3 & DiffPC4.5	H & V	-	Differential Privacy build into the creation of a DT.	Trade-off between privacy parameter and accuracy. Different stopping rules could be assessed.
Blum *et al.* [43]	DT & more	SuLQ ID3	H & V	Honest-but-curious	Insert noise into queries to Data	
Lindel & Pinkas [51]	DT	Private ID3$_\delta$	H & V	Honest-but-curious	Two-Party Union, using Oblivious Transfer and Circuits	Manageable Overhead due to most computations being done locally
Vaidya *et al.* [52]	DT	-	V	Honest-but-curious	Two or more privacy-preserving DT setup with a black-box, individual participants need to be online to query model.	General framework for distributed classification. Aims to be a proof of concept. Can take days to complete
Sav *et al.* [53]	NN	POSEIDON	H	Honest-but-curious & participants	The model stays encrypted using homomophic encryption & MPC	Linearly scaling communicational overhead
Wagh *et al.* [54]	NN	SecureNN	H	Honest-but-curious & malicious	Data split using secret shares between 3 (independent) servers, inference through MLaaS, Homomorphic MPC	Fast and accurate NN training, fast enough for CNN applications. Communication is biggest overhead

TABLE XIV: Privacy-Preserving & Distributed Algorithms, H = Horizontally partitioned data, V = Vertically partitioned data.