2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00023

Feature Inference Attack on Model Predictions in Vertical Federated Learning

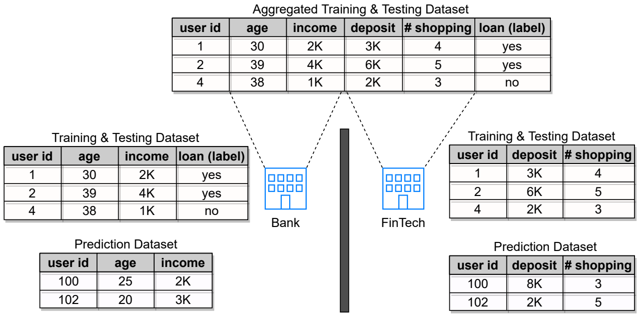
Xinjian Luo, Yuncheng Wu*, Xiaokui Xiao, Beng Chin Ooi
National University of Singapore
{*xinjluo, wuyc, xiaoxk, ooibc*}*@comp.nus.edu.sg*

***Abstract*—Federated learning (FL) is an emerging paradigm for facilitating multiple organizations' data collaboration without revealing their private data to each other. Recently, vertical FL, where the participating organizations hold the same set of samples but with disjoint features and only one organization owns the labels, has received increased attention. This paper presents several feature inference attack methods to investigate the potential privacy leakages in the model prediction stage of vertical FL. The attack methods consider the most stringent setting that the adversary controls only the trained vertical FL model and the model predictions, relying on no background information of the attack target's data distribution. We first propose two specific attacks on the logistic regression (LR) and decision tree (DT) models, according to individual prediction output. We further design a general attack method based on multiple prediction outputs accumulated by the adversary to handle complex models, such as neural networks (NN) and random forest (RF) models. Experimental evaluations demonstrate the effectiveness of the proposed attacks and highlight the need for designing private mechanisms to protect the prediction outputs in vertical FL.**

***Index Terms*—vertical federated learning, feature inference attack, model prediction, privacy preservation**

I. Introduction

Recent years have witnessed a growing interest in exploiting data from distributed sources of multiple organizations, for designing sophisticated machine learning (ML) models and providing better customer service and acquisition. However, proprietary data cannot be directly shared for two reasons. On the one hand, the usage of user's sensitive data is compelled to abide by standard privacy regulations or laws, e.g., GDPR [1] or CCPA [2]. On the other hand, the data is a valuable asset to the organizations for maintaining a competitive advantage in business, which should be highly protected.



Aggregated Training & Testing Dataset

user id	age	income	deposit	# shopping	loan (label)
1	30	2K	3K	4	yes
2	39	4K	6K	5	yes
4	38	1K	2K	3	no

Training & Testing Dataset

user id	age	income	loan (label)
1	30	2K	yes
2	39	4K	yes
4	38	1K	no

Training & Testing Dataset

user id	deposit	# shopping
1	3K	4
2	6K	5
4	2K	3

Prediction Dataset

user id	age	income
100	25	2K
102	20	3K

Prediction Dataset

user id	deposit	# shopping
100	8K	3
102	2K	5

Fig. 1: Example of vertical federated learning

Federated learning (FL) [3, 4, 5, 6, 7] is an emerging paradigm for data collaboration that enables multiple data owners (i.e., parties) to jointly build an ML model and serve new requests without revealing their private data to each other. FL can be categorized into different scenarios according to the data partitioning. In this paper, we consider vertical FL [4, 8, 9, 10, 11, 12] where the participating parties hold the same set of samples while each party only has a disjoint subset of features. Vertical FL has been demonstrated effective in many real-world applications and received increased attention in organizations or companies [10, 11]. Fig. 1 illustrates a digital banking example. A bank aims to build an ML model to evaluate whether to approve a user's credit card application by incorporating more features from a Fintech company. The bank holds features of 'age' and 'income' while the Fintech company holds features of 'deposit' and 'average online shopping times'. Only the bank owns the label information in the *training dataset* and *testing dataset*, i.e., the ground truth that indicates whether an application should be approved. We refer to the party with the label as the *active party* and the other parties as *passive parties*. To train a vertical FL model, the parties iteratively exchange certain intermediate results in a secure manner until obtaining a jointly trained model [4]. Finally, the trained model will be released to the parties for justifying the model's effectiveness and interpretability.

After obtaining the trained model, the parties utilize it to collaboratively compute model predictions for new samples in the *prediction dataset*, for example, new credit card applicants in Fig. 1. In practice, the prediction outputs will be revealed to the active party for making further decisions. Though cryptographic techniques, e.g., partially homomorphic encryption [13] and secure multiparty computation [14], can be applied to ensure that no intermediate information during the computation is disclosed [4], the prediction outputs must contain information of the parties' sensitive data because they are computed upon the private data. Therefore, in this paper, we aim to investigate the key question: *how much information about a passive party's feature values can be inferred by the active party from the model predictions in vertical FL?*

Although a number of feature inference attacks are proposed against machine learning [15, 16, 17, 18, 19, 20, 21], none of them are applicable to our problem. Many existing attacks [15, 16, 17, 18] aim to infer a participating party's feature values in horizontal FL [3, 5, 6], where the parties have the same features but with different samples. However, these

*Corresponding author.

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00023

attacks greatly rely on the model gradients exchanged during the training process, which unintentionally memorize sensitive information of the training samples. Once the model gradients are safely protected, e.g., using cryptographic techniques [5, 6], these attacks would be invalid. Also, for the inference attack in the prediction stage (i.e., considered in this paper), the FL model is not aware of the predicting samples beforehand. Thus, the memorization nature used in these attacks does not work. Another thread of feature inference attacks [19, 20, 21] assumes that the adversary could obtain some auxiliary statistics or marginal data distribution about the unknown features. This assumption is relatively strong, as it is challenging in practice for the active party to collect such information from other parties in vertical FL.

In this paper, we study the privacy leakage problem in the prediction stage of vertical FL, by presenting several feature inference attacks based on model predictions. Our attack methods do not rely on any intermediate information during the computation of prediction outputs. For example, we assume that the parties can jointly run a secure protocol such that only the prediction outputs are released to the active party and nothing else. Besides, unlike previous work [19, 20, 22], the proposed methods need no statistics or distributions of the attack target's data. Therefore, we consider the most stringent setting that the active party (i.e., the adversary) controls only the trained vertical FL model and the model predictions.

From the experiments, we observe that those model predictions can leak considerable information about the features held by the passive parties, especially when specific conditions are satisfied, e.g., the number of classes in the classification is large or the adversary's features and the passive parties' features are highly correlated. In light of these observations, we design several defense strategies and incorporate them into our *Falcon*[1] (federated learning with privacy protection) system to safeguard against such potential risks. Specifically, we make the following contributions.

- We formulate the problem of feature inference attack on model predictions in vertical FL, where the active party attempts to infer the feature values of new samples belong to the passive parties. To the best of our knowledge, this is the first work that investigates this form of privacy leakage in vertical FL.
- We propose two specific attacks on the logistic regression (LR) and decision tree (DT) models when the adversary only has an individual prediction output. These attacks are straightforward to initiate and can achieve high accuracy when certain conditions are satisfied.
- We further design a general attack which learns the correlations between the adversary's features and the attack target's features using multiple predictions accumulated by the adversary. This attack can handle more complex models, such as neural networks (NN) and random forest (RF).
- We implement the proposed attacks and conduct extensive evaluations on both real-world and synthetic datasets. The results demonstrate the effectiveness of our attacks and highlight the need for designing defense mechanisms to mitigate the privacy risks arising from federated model predictions. We also analyze and suggest several potential countermeasures against these attacks.

II. Preliminaries

A. Machine Learning

In this paper, we focus on supervised classification learning, which builds a model from a labeled training dataset consisting of a set of samples [16]. Given a training dataset $\mathcal{D}_{\text{train}}$ with n samples $\boldsymbol{x}^t (t \in \{1, \cdots, n\})$ each containing d features and the corresponding output label y^t, to learn the parameters $\boldsymbol{\theta}$, the training algorithm optimizes a loss function, i.e.,

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{t=1}^{n} \ell(f(\boldsymbol{x}^t, \boldsymbol{\theta}), y^t) + \Omega(\boldsymbol{\theta}) \qquad (1)$$

where $\ell(f(\boldsymbol{x}^t, \boldsymbol{\theta}), y^t)$ denotes the loss of sample $\boldsymbol{x}^t$ with label y^t, and $\Omega(\boldsymbol{\theta})$ is the regularization term that penalizes model complexity to avoid overfitting. After obtaining the trained model parameters $\boldsymbol{\theta}$, we can compute a vector of confidence scores $\boldsymbol{v} = (v_1, \cdots, v_c)$ for any input sample $\boldsymbol{x}$, where each score represents the probability of $\boldsymbol{x}$ belonging to a class label, and c is the number of classes. Consequently, the class label of $\boldsymbol{x}$ is determined by the highest confidence score.

Logistic regression (LR) model prediction. In binary LR classification with two classes, the confidence score vector is $\boldsymbol{v} = (f(\boldsymbol{x}, \boldsymbol{\theta}), 1 - f(\boldsymbol{x}, \boldsymbol{\theta}))$, where $f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x})$ and $\sigma(x) = (1 + e^{-x})^{-1}$, representing the probability that the input sample is classified into the first class. In multi-class LR classification (i.e., $c > 2$), a typical method is multinomial classification, which trains a linear regression model $\boldsymbol{\theta}^{(k)}$ for each class $k \in \{1, \cdots, c\}$ and applies a softmax function on the c linear regression predictions $f(\boldsymbol{x}, \boldsymbol{\theta}^{(1)}), \cdots, f(\boldsymbol{x}, \boldsymbol{\theta}^{(c)})$. As a consequence, the output of the softmax function is the confidence score vector, i.e., the probabilities of the c classes.

Neural networks (NN) model prediction. The neural network model is a popular architecture in deep learning. In NN, f is typically composed of an input layer, an output layer, and a sequence of hidden layers with non-linear transformations from the input to the output. The model parameters $\boldsymbol{\theta}$ denote the weights for each transformation. In the prediction stage, the results of the output layer are the confidence scores.

Tree-based model prediction. The decision tree (DT) model consists of internal nodes and leaf nodes, where each internal node describes a branching threshold *w.r.t.* a feature, and each leaf node represents a class label. When predicting an input sample $\boldsymbol{x}$, a sequence of branching operations are executed based on comparisons of $\boldsymbol{x}$'s features and the thresholds, until a leaf node is reached. The prediction output is the class label on that leaf node. Note that the prediction process is deterministic. Therefore, the confidence score for the predicted class is 1, and otherwise 0.

[1] https://www.comp.nus.edu.sg/~dbsystem/fintech-Falcon/

We can use ensemble models to further improve predictive performance, e.g., random forest (RF). Essentially, the RF model is composed of a set of independent DT models. In the prediction stage, each tree predicts a class label, and the final classification is calculated by majority voting from all trees. As a result, the prediction output of the RF model also includes a vector of confidence scores, where each element v_k of class $k \in \{1, \cdots, c\}$ is the fraction of trees that predict k.

B. Vertical Federated Learning

Vertical FL enables an active party (e.g., the bank in Fig. 1) to enrich his business data by incorporating more diverse features of users from one or several passive parties (e.g., the Fintech company in Fig 1). From which the active party could improve his model accuracy to provide better customer service. Meanwhile, the passive parties could benefit from a pay-per-use model [23, 24] for their contributions to the model training and model prediction stages.

In the prediction stage of vertical FL, a model prediction is usually initiated by the active party who sends a prediction request to other parties, along with an input sample id (e.g., 100 or 102 in Fig. 1). The passive parties then prepare the corresponding feature values of that sample, and all parties jointly run a protocol to compute the prediction output. The protocol can be secure enough to protect each party's sensitive data, such that only the prediction output is revealed to the active party for making further decisions, and no intermediate information during the computation is disclosed [4].

III. Problem Statement

A. System Model

We consider a set of m distributed parties (or data owners) $\{P_1, \cdots, P_m\}$ who train a vertical FL model by consolidating their respective datasets. After obtaining the trained vertical FL model parameters $\boldsymbol{\theta}$, the parties collaboratively make predictions on their joint prediction dataset $\mathcal{D}_{\text{pred}} = \{D_1, \cdots, D_m\}$. Each row in the dataset corresponds to a sample, and each column corresponds to a feature. Let n be the number of samples and d_i be the number of features in D_i, where $i \in \{1, \cdots, m\}$. We denote $D_i = \{\boldsymbol{x}_i^t\}_{t=1}^n$ where $\boldsymbol{x}_i^t$ represents the t-th sample of D_i. In this paper, we consider the supervised classification task and denote the number of classes by c. Table I summarizes the frequently used notations.

In vertical FL, the datasets $\{D_1, \cdots, D_m\}$ share the same sample ids but with disjoint subsets of features [7]. In particular, we assume that the parties have determined and aligned their common samples using private set intersection techniques [25] without revealing the samples not in the intersection.

B. Threat Model

We consider the semi-honest model [26] where every party follows the protocol exactly as specified, but may try to infer other parties' private information based on the messages received. Specifically, this paper focuses on the situation that the active party is the adversary. The active party may also collude with other passive parties to infer the private feature values of a set of target passive parties. The strongest notion is that $m - 1$ parties collude (including the active party) to infer the features of the remaining passive party. In addition, we assume that the active party has no background information on the passive parties' data distribution. However, he may know the passive parties' feature names, types, and value ranges; this is reasonable because the active party often needs this information to justify the effectiveness of the trained model.

TABLE I: Summary of notations

Notation	Description
d	the number of total features
c	the number of classes in the classification
$\boldsymbol{x}$	an input sample in the prediction stage
$\boldsymbol{v}$	the prediction output, i.e., confidence scores of $\boldsymbol{x}$
P_{adv}	the adversary (the active party and several passive parties)
P_{target}	the attack target (the remaining passive parties)
d_{adv}	the number of features hold by P_{adv}
d_{target}	the number of features hold by P_{target}
$\boldsymbol{x}_{\text{adv}}$	the feature values of $\boldsymbol{x}$ hold by P_{adv}
$\boldsymbol{x}_{\text{target}}$	the feature values of $\boldsymbol{x}$ hold by P_{target}
$\boldsymbol{\theta}$	the parameters of the trained vertical FL model
$\boldsymbol{\theta}_G$	the parameters of the generator model

C. Feature Inference Attack on Model Predictions

Given the vertical FL model parameters $\boldsymbol{\theta}$, the parties can jointly make predictions on new input samples, whose feature values are distributed among m parties. As described in Section II-A, the prediction output consists of a vector of confidence scores, i.e., $\boldsymbol{v} = (v_1, \cdots, v_c)$. We assume that $\boldsymbol{\theta}$ and $\boldsymbol{v}$ are revealed to the active party as this information is essential for the active party to make correct decisions.

Let $\boldsymbol{x}$ be an input sample for prediction. Without loss of generality, the m parties can be abstracted into two parties: the adversary P_{adv}, which is composed of the active party and a subset of colluding passive parties; and the attack target P_{target}, which is composed of the remaining passive parties. Let $\boldsymbol{x} = (\boldsymbol{x}_{\text{adv}}, \boldsymbol{x}_{\text{target}})$, such that $\boldsymbol{x}_{\text{adv}}$ and $\boldsymbol{x}_{\text{target}}$ denote the feature values held by P_{adv} and P_{target}, respectively; and d_{adv} and d_{target} denote the number of features held by P_{adv} and P_{target} accordingly. As a consequence, the setting for our feature inference attack is as follows. The adversary P_{adv} is given the vertical FL model parameters $\boldsymbol{\theta}$, the prediction output $\boldsymbol{v}$, and the feature values $\boldsymbol{x}_{\text{adv}}$ that belongs to himself. P_{adv}'s goal is to infer the feature values of P_{target}, i.e.,

$$\hat{\boldsymbol{x}}_{\text{target}} = \mathcal{A}(\boldsymbol{x}_{\text{adv}}, \boldsymbol{v}, \boldsymbol{\theta}) \tag{2}$$

where $\hat{\boldsymbol{x}}_{\text{target}}$ is the inferred feature values and $\mathcal{A}$ is an attack algorithm executed by P_{adv}.

IV. Attack Based on Individual Prediction

In this section, we present an equality solving attack on the logistic regression (LR) model and a path restriction attack on the decision tree (DT) model based on an individual model prediction, which means that the adversary can initiate an attack once obtaining the prediction output of a sample.

A. Equality Solving Attack

As described in Section II-A, the LR prediction of a sample $\boldsymbol{x}$ is computed by a deterministic function $f(\boldsymbol{x}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the model parameters known to the active party (i.e., the adversary). Therefore, given the prediction $\boldsymbol{v}$, the adversary P_{adv} can construct a set of equations, from which P_{adv} could infer the feature values held by P_{target}. We discuss the binary LR classification and multi-class LR classification separately.

Binary LR prediction. We denote the model parameters as $\boldsymbol{\theta} = (\boldsymbol{\theta}_{\text{adv}}, \boldsymbol{\theta}_{\text{target}})$, where $\boldsymbol{\theta}_{\text{adv}}$ and $\boldsymbol{\theta}_{\text{target}}$ are the weights corresponding to the features owned by P_{adv} and P_{target}, respectively. Let $\boldsymbol{v}$ be the prediction output of a given sample $\boldsymbol{x} = (\boldsymbol{x}_{\text{adv}}, \boldsymbol{x}_{\text{target}})$. For binary LR classification, there is only one meaningful confidence score in $\boldsymbol{v}$, e.g., v_1 for the first class, and that for the second class is $1 - v_1$. Given v_1 and the adversary's own feature values $\boldsymbol{x}_{\text{adv}}$, it is straightforward for P_{adv} to create an equation with $\boldsymbol{x}_{\text{target}}$ as the variables, i.e.,

$$\sigma(\boldsymbol{x}_{\text{adv}} \cdot \boldsymbol{\theta}_{\text{adv}} + \boldsymbol{x}_{\text{target}} \cdot \boldsymbol{\theta}_{\text{target}}) = v_1 \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function. Obviously, if there is only one unknown feature, i.e., $d_{\text{target}} = 1$, then the equation has only one solution, which means the unknown feature value $\boldsymbol{x}_{\text{target}}$ can be inferred precisely.

Multi-class LR prediction. For multi-class LR classification, as mentioned in Section II-A, there are c linear regression models. Let $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(1)}, \cdots, \boldsymbol{\theta}^{(c)})$ be the parameters for these c models, respectively. To initiate the feature inference attack, the adversary aims to construct the following linear equations for $k \in \{1, \cdots, c\}$.

$$\boldsymbol{x}_{\text{adv}} \cdot \boldsymbol{\theta}^{(k)}_{\text{adv}} + \boldsymbol{x}_{\text{target}} \cdot \boldsymbol{\theta}^{(k)}_{\text{target}} = z_k \tag{4}$$

where z_k is the output of the k-th linear regression model. However, P_{adv} only knows the confidence score vector $\boldsymbol{v} = (v_1, \cdots, v_c)$ such that

$$v_k = \frac{\exp(z_k)}{\sum_{k'} \exp(z_{k'})}. \tag{5}$$

Also, it is impossible for P_{adv} to revert $z_1, \cdots, z_c$ from $v_1, \cdots, v_c$ because there are multiple solutions to Eqn (5). Nevertheless, we observe that an important correlation exists between two confidence scores. Specifically, we can transform Eqn (5) into $\ln v_k = z_k - \ln(\sum_{k'} \exp(z_{k'}))$. Note that the second term in the right hand of this equation is the same for all $k \in \{1, \cdots, c\}$. As a result, P_{adv} can subtract $\ln v_k$ by another confidence score, say $\ln v_{k'}$, obtaining:

$$\ln v_k - \ln v_{k'} = z_k - z_{k'}. \tag{6}$$

This property allows P_{adv} to obtain $c - 1$ linear equations by subtracting two adjacent equations in Eqn (4), i.e.,

$$\boldsymbol{x}_{\text{adv}}(\boldsymbol{\theta}^{(k)}_{\text{adv}} - \boldsymbol{\theta}^{(k+1)}_{\text{adv}}) + \boldsymbol{x}_{\text{target}}(\boldsymbol{\theta}^{(k)}_{\text{target}} - \boldsymbol{\theta}^{(k+1)}_{\text{target}}) = a'_k \tag{7}$$

where $a'_k = \ln v_k - \ln v_{k+1}$ and $k \in \{1, \cdots, c-1\}$. Consequently, if the number of unknown features $d_{\text{target}} \leq c - 1$, then there will be only one solution for $\boldsymbol{x}_{\text{target}}$, which can be inferred exactly.

Attack method. Based on the discussion for both binary LR and multi-class LR, we observe that when a specific condition is satisfied, i.e., $d_{\text{target}} \leq c - 1$, P_{adv} can infer $\boldsymbol{x}_{\text{target}}$ precisely. Due to that Eqn (3) and Eqn (7) are linear equations, we can rewrite them into matrix form, i.e., $\boldsymbol{\Theta}_{\text{target}} \boldsymbol{x}_{\text{target}} = \boldsymbol{a}$, where the dimension of $\boldsymbol{\Theta}_{\text{target}}$ is $(c-1) \times d_{\text{target}}$, and $\boldsymbol{x}_{\text{target}}$ and $\boldsymbol{a}$ are column vectors with size d_{target} and $c - 1$, respectively. In particular, for binary LR, $\boldsymbol{\Theta}_{\text{target}}$ equals to $\boldsymbol{\theta}_{\text{target}}$ and $\boldsymbol{a}$ is $\sigma^{-1}(v_1) - \boldsymbol{x}_{\text{adv}} \cdot \boldsymbol{\theta}_{\text{adv}}$. While for multi-class LR, $\boldsymbol{\Theta}_{\text{target}}$ is composed of $c-1$ vectors $\boldsymbol{\theta}^{(k)}_{\text{target}} - \boldsymbol{\theta}^{(k+1)}_{\text{target}}$ for $k \in \{1, \cdots, c-1\}$ and $\boldsymbol{a}$ is derived from Eqn (7) by substituting the known values. Given this matrix representation, the adversary can solve the target feature values easily by $\hat{\boldsymbol{x}}_{\text{target}} = \boldsymbol{\Theta}^{+}_{\text{target}} \boldsymbol{a}$, where $\boldsymbol{\Theta}^{+}_{\text{target}}$ is the pseudo-inverse matrix [27] of $\boldsymbol{\Theta}_{\text{target}}$. When $d_{\text{target}} \geq c$, although there are infinite solutions to Eqn (7), the solution $\hat{\boldsymbol{x}}_{\text{target}}$ constructed by $\boldsymbol{\Theta}^{+}_{\text{target}} \boldsymbol{a}$, which minimizes $||\boldsymbol{\Theta}_{\text{target}} \hat{\boldsymbol{x}}_{\text{target}} - \boldsymbol{a}||_2$ and satisfies $||\hat{\boldsymbol{x}}_{\text{target}}||_2 \leq ||\boldsymbol{x}_{\text{target}}||_2$ for all solutions (we refer the interested readers to [27] for more details), can still give a good estimation of $\boldsymbol{x}_{\text{target}}$. We will experimentally demonstrate this in Section VI.

Example 1. *Suppose the Bank in Fig. 1 tries to infer a user's unknown feature values that belong to the FinTech company. To illustrate the equality solving attack, we assume that there are 3 classes and the trained model parameters are* $\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\theta}^{(1)} \\ \boldsymbol{\theta}^{(2)} \\ \boldsymbol{\theta}^{(3)} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.0002 & 0.0005 & 0.09 \\ 0.06 & 0.0005 & 0.0002 & 0.08 \\ 0.01 & 0.0001 & 0.0004 & 0.05 \end{bmatrix}$. *Suppose there is an input sample* $\boldsymbol{x} = (25, 2K, 8K, 3)$*, and the predicted confidence score vector is* $\boldsymbol{v} = (0.867, 0.084, 0.049)$. P_{adv} *has the former two features 'age=25' and 'income=2K'. Then, from Eqn (6)-(7), he computes* $\boldsymbol{\theta}^{(1)}_{adv} - \boldsymbol{\theta}^{(2)}_{adv} = (0.02, -0.0003)$, $\boldsymbol{\theta}^{(2)}_{adv} - \boldsymbol{\theta}^{(3)}_{adv} = (0.05, 0.0004)$, $\boldsymbol{\theta}^{(1)}_{target} - \boldsymbol{\theta}^{(2)}_{target} = (0.0003, 0.01)$, $\boldsymbol{\theta}^{(2)}_{target} - \boldsymbol{\theta}^{(3)}_{target} = (-0.0002, 0.03)$, $a'_1 = 2.334$*, and* $a'_2 = 0.539$*. Thus,* P_{adv} *can estimate* $\hat{\boldsymbol{x}}_{target} = (8011.8, 3.046)$*, where the loss is from precision truncation during the computations.*

B. Path Restriction Attack

For the DT model, the prediction output includes only the predicted class, with a confidence score of 1. We propose a specific attack called *path restriction attack*, by which the adversary can restrict the possible prediction paths in the tree model based on the predicted class and his own feature values.

Example 2. *Fig. 2 illustrates the basic idea of this attack given Fig. 1. The adversary* P_{adv} *has partial feature values: 'age=25' and 'income=2K'. Then* P_{adv} *can restrict the possible prediction paths from 5 (the total number of prediction paths) to 2 (with blue arrows). Besides, if assuming that the predicted class of this sample is 1,* P_{adv} *can identify the real prediction path (with red arrows) and correctly infer that* P_{target}*'s deposit feature value of this sample is larger than 5K.*

Generally, let n_p be the total number of prediction paths, the adversary can restrict the candidate prediction paths to n_r paths by comparing his own feature values with the branching thresholds in the tree model and checking if the leaf label of a path matches the ground-truth predicted class. Consequently,

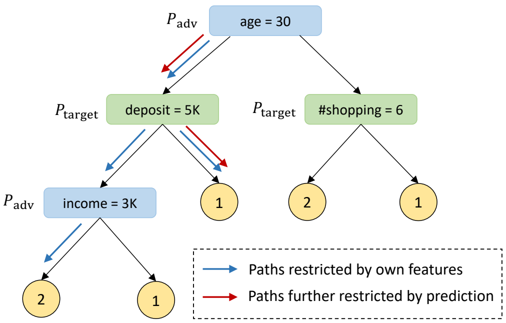


Fig. 2: Example of path restriction attack. Assume that P_{adv}'s feature values of an input sample is: 'age=25' and 'income=2K', while the predicted class is 1.

the adversary uniformly and randomly selects a path from the n_r paths and checks the unknown feature values belong to which branch compared to the branching thresholds.

Algorithm 1 summarizes this attack method. Let n_f be the number of tree nodes in the full binary tree. The adversary P_{adv} first initializes an indicator vector $\boldsymbol{\beta}$ of size n_f with 0 (line 1). This vector is used for recording the tree nodes that may be evaluated from the perspective of P_{adv}. Then P_{adv} initializes a queue Q with the root node (whose index is 0) and sets $\beta_0 = 1$ because the root node must be evaluated for any prediction (lines 2-3). Next, P_{adv} iteratively pops a node from Q and checks the following conditions until Q is empty (lines 4-14). If the feature on the current node belongs to P_{adv}, P_{adv} updates the corresponding indicators of its child nodes in $\boldsymbol{\beta}$ according to the comparison result between his feature value and the branching threshold on that node (lines 6-10). Otherwise, the indicators of the child nodes are the same as the indicator of the current node (line 12). For example, in Fig. 2, given the branching condition 'age=30' on the root node ($i = 0$) and the feature value 'age=25' of the input sample, P_{adv} can update $\beta_{2i+1} = \beta_i = 1$ and $\beta_{2i+2} = 0$ as the prediction goes into the left branch. After that, if the current node is not a leaf node, P_{adv} pushes its child nodes into Q for iteratively updating. As a result, $\boldsymbol{\beta}$ can be computed. For example, $\boldsymbol{\beta} = (1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0)$ given Fig. 2. In the following, P_{adv} computes another indicator vector $\boldsymbol{\alpha}$ with size n_f, such that the elements of leaf nodes with label k are set to 1 and the others are set to 0 (line 15). For example, $\boldsymbol{\alpha} = (0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0)$ in Fig. 2. Finally, P_{adv} updates $\boldsymbol{\beta}$ by element-wise multiplication using $\boldsymbol{\alpha}$, and the restricted prediction paths are those elements with 1 in $\boldsymbol{\beta}$ (lines 16-17). In Fig. 2, there is only one element is 1 (i.e., node index 4) in the updated $\boldsymbol{\beta}$, and the corresponding prediction path can be identified (i.e., with red arrows). The complexity of Algorithm 1 is $O(n_f)$ as the adversary needs to traverse the tree model to compute the indicator vectors.

V. Attack Based on Multiple Predictions

So far, we identify two specific attacks based on individual model prediction. However, these attacks could hardly be applied to complex models, such as neural networks (NN) and random forest (RF). For the NN model, there are a number of hidden layers for non-linear transformations, making it difficult to solve the equations by the equality solving attack. Even though the adversary could utilize the maximum a posteriori (MAP) method (e.g., [20]) to find a plausible solution, the attack accuracy would be undesirable because the solution space to the unknown features in the NN model is huge and irregular. For the RF model, given the prediction output (i.e., the confidence scores), the number of candidate tree combinations might be too large. For example, if there are 100 trees in the RF model and a prediction output with two classes is $\boldsymbol{v} = (0.4, 0.6)$, then the adversary needs to consider C_{100}^{40} combinations of the trees, which is computationally expensive.

```
Algorithm 1: Path restriction attack
  Input: $\boldsymbol{x}_{\text{adv}}$: adversary's feature values, $\boldsymbol{\theta}$: model parameters,
         $k$: predicted class
  Output: $\mathcal{P}$: possible prediction paths
1 $\boldsymbol{\beta} = \mathbf{0}$ // initialize indicator vector
2 $Q \leftarrow$ root node (index 0)
3 $\beta_0 = 1$
4 while $Q$ is not empty do
5     node $i \leftarrow Q$.pop()
6     if node $i$ belongs to the adversary then
7         if adversary's feature value $\leq$ threshold value then
8             $\beta_{2i+1} = \beta_i$, $\beta_{2i+2} = 0$
9         else
10            $\beta_{2i+1} = 0$, $\beta_{2i+2} = \beta_i$
11    else
12        $\beta_{2i+1} = \beta_{2i+2} = \beta_i$
13    if node $i$ is not leaf node then
14        $Q$.push$(2i + 1)$, $Q$.push$(2i + 2)$
15 $\boldsymbol{\alpha} \leftarrow$ compute the indicator vector, such that the elements of
     leaf nodes with label $k$ are set to 1 and otherwise 0
16 $\boldsymbol{\beta} \leftarrow$ element-wise multiplication of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$
17 $\mathcal{P} \leftarrow$ find the paths that $\beta' = 1$ ($\forall \beta' \in \boldsymbol{\beta}$)
18 return $\mathcal{P}$
```

To address the above limitations, we design a general feature inference attack, called *generative regression network* (GRN), based on multiple model predictions. The rationale that the adversary can rely on multiple predictions is that the active party can easily collect this information by observing model predictions of new samples in the long term, for example, in a week or a month, as long as the vertical FL model is unchanged. These accumulated data could be used for initiating a more sophisticated inference attack.

Fig. 3 gives an overview of this attack. The basic idea is to figure out the overall correlations between the adversary's own features and the attack target's unknown features. Upon this, the problem of inferring the unknown feature values is equivalent to the problem of generating new values $\hat{\boldsymbol{x}}_{\text{target}}$ to match the decisions of vertical FL model, where $\hat{\boldsymbol{x}}_{\text{target}}$ follows a probability distribution determined by the adversary's known feature values and the feature correlations. To learn such a probability distribution, we build a generator model (in green color in Fig. 3) which takes the adversary's known feature

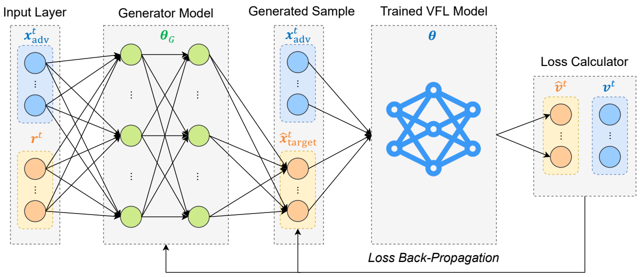


Fig. 3: Illustration of generative regression network training

values and a set of random variables as inputs (as depicted in the input layer in Fig. 3) and produces an estimation of the unknown feature values. The estimated values and the known feature values are combined into a *generated sample*. Consequently, the generator model is trained by minimizing the loss between the predictions of the generated samples and the ground-truth predictions.

Example 3. *Suppose that the Bank in Fig. 1 knows feature values* $\boldsymbol{x}^t_{adv} = (age\ 25, income\ 2K)$*, and the prediction output* $v^t = 0.7$ *if there are two classes. To infer the other two feature values, the Bank first feeds* $\boldsymbol{x}^t_{adv}$ *together with some random values* $(25, 2K, r^t_1, r^t_2)$ *into the generator to obtain a preliminary result, e.g.* $\hat{\boldsymbol{x}}^t_{target} = (2K, 2)$*; then,* $\boldsymbol{x}^t_{adv}$ *and* $\hat{\boldsymbol{x}}^t_{target}$ *are concatenated and input into the federated model to get a simulated output* $\hat{v}^t = 0.5$*; the loss function* $\ell(\hat{v}^t, v^t)$ *is then calculated and back-propagated to the generator. With multiple predictions, the generator can learn the data distribution of the target features.*

Note that the random vector $\boldsymbol{r}^t$ is an indispensable component of GRN. First, it acts as a regularizer, encouraging the generator to capture more randomness arising from $\boldsymbol{x}^t_{\text{adv}}$. Second, in different epochs, the $\boldsymbol{x}^t_{\text{adv}}$ of a specific sample can be concatenated with different $\boldsymbol{r}^t$ into different generator inputs, which derive various gradient directions during the back-propagation and thus lead to a better estimation of $\boldsymbol{x}^t_{\text{target}}$ with high probabilities. In the experiments, we found that a generator with random inputs can reduce the reconstruction error by 20% compared to those without random inputs.

Unlike existing feature inference attacks, our attack method considers the most stringent case: it relies on no intermediate information disclosed during the computation (required in [16, 17, 18]) and no background information of the attack target's data distribution (such as statistics or marginal feature distribution required in [19, 20, 21]). In particular, the adversary only needs a set of model predictions and the vertical FL model for initiating the attack.

A. Generative Regression Network

We assume that the adversary P_{adv} has collected prediction outputs of n samples in $\mathcal{D}_{\text{pred}}$, where $\mathcal{D}_{\text{pred}} = (D_{\text{adv}}, D_{\text{target}})$ such that the t-th ($t \in \{1, \cdots, n\}$) sample is represented by $\boldsymbol{x}^t = (\boldsymbol{x}^t_{\text{adv}}, \boldsymbol{x}^t_{\text{target}})$. Let $V = (\boldsymbol{v}^1, \cdots, \boldsymbol{v}^n)$ be the corresponding n prediction outputs. The adversary's objective is to train a generator model, say $\boldsymbol{\theta}_G$, such that given the known feature values of the t-th sample $\boldsymbol{x}^t_{\text{adv}}$ and a random value vector $\boldsymbol{r}^t$ with size d_{target}, the generator outputs the corresponding estimation $\hat{\boldsymbol{x}}^t_{\text{target}}$ of the target party's $\boldsymbol{x}^t_{\text{target}}$. To train the model, P_{adv} can apply the mini-batch stochastic gradient descent method, where the training dataset is composed of D_{adv} and V. The objective function is as follows:

$$\min_{\boldsymbol{\theta}_G} \frac{1}{n}\sum_{t=1}^{n} \ell(f(\boldsymbol{x}^t_{\text{adv}}, f_G(\boldsymbol{x}^t_{\text{adv}}, \boldsymbol{r}^t; \boldsymbol{\theta}_G); \boldsymbol{\theta}), \boldsymbol{v}^t) + \Omega(f_G) \quad (8)$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_G$ are the parameters of the vertical FL model and the generator model, respectively. Moreover, f_G denotes $\hat{\boldsymbol{x}}^t_{\text{target}}$, i.e., the output of the generator, and f denotes the output of the vertical FL model given the generated sample (concatenated by $\boldsymbol{x}^t_{\text{adv}}$ and $\hat{\boldsymbol{x}}^t_{\text{target}}$). Besides, $\Omega(\cdot)$ is a regularization term of the generated unknown feature values $\{\hat{\boldsymbol{x}}^t_{\text{target}}\}^n_{t=1}$. For example, we penalize the generator model when the variance of $\{\hat{\boldsymbol{x}}^t_{\text{target}}\}^n_{t=1}$ is too large, preventing from generating meaningless samples. Nevertheless, the variance is computed based on the generated values, thus no prior information is needed by the adversary.

Algorithm 2 presents the training method for GRN. Specifically, in each iteration of each epoch, P_{adv} first selects a batch of samples $B = (\boldsymbol{x}^1_{\text{adv}}, \cdots, \boldsymbol{x}^{|B|}_{\text{adv}})$ from D_{adv} and initializes a set of $|B|$ random vectors $R = (\boldsymbol{r}^1, \cdots, \boldsymbol{r}^{|B|})$. The size of each random vector is the number of features held by the attack target P_{target}. Then for the t-th sample in B, P_{adv} feeds $\boldsymbol{x}^t_{\text{adv}}$ along with the corresponding $\boldsymbol{r}^t$ into the generator, obtaining $\hat{\boldsymbol{x}}^t_{\text{target}}$. Next, P_{adv} concats $\boldsymbol{x}^t_{\text{adv}}$ with $\hat{\boldsymbol{x}}^t_{\text{target}}$ to obtain a complete generated sample and makes a prediction using the vertical FL model, resulting in a prediction output $\hat{\boldsymbol{v}}^t$. As a result, the loss of this generated sample can be calculated by a loss function $\ell(\cdot, \cdot)$, e.g., MSE, with the ground-truth prediction output $\boldsymbol{v}^t$. After obtaining the losses of all samples in B, the adversary back-propagates the aggregated loss to update the parameters of the generator model $\boldsymbol{\theta}_G$. Finally, the adversary can obtain the trained generator model after all the training epochs.

After obtaining $\boldsymbol{\theta}_G$, the adversary can use it to infer the unknown feature values D_{target} in $\mathcal{D}_{\text{pred}}$. Specifically, for any sample $\boldsymbol{x}$ in $\mathcal{D}_{\text{pred}}$, the adversary generates a random vector $\boldsymbol{r}$ and directly computes $f_G(\boldsymbol{x}_{\text{adv}} \cup \boldsymbol{r}, \boldsymbol{\theta}_G)$ as the inferred feature values. Notice that the samples to be attacked are exactly the samples for training the generator model. This property ensures that the adversary can accumulate as many model predictions as possible to train the generator and improve the attack performance. We will experimentally show the effect of the number of model predictions in Section VI.

The GRN attack method is general because it takes the trained vertical FL model as a black-box, as long as that model's objective function is differentiable, such that the prediction loss could be back-propagated to the generator model. Thus, this attack works for both LR and NN models.

B. Adopt GRN Attack on the Random Forest Model

Let W be the number of trees in the RF model. When predicting a sample, each tree produces a predicted class. The prediction output $\boldsymbol{v} = (v_1, \cdots, v_c)$ represents the fraction of trees that predict a label for each class. As discussed before,

Algorithm 2: Generative regression network training

```
Input: $\{\boldsymbol{x}^t_{\text{adv}}\}_{t=1}^n$: adversary's feature values, $\boldsymbol{\theta}$: parameters
       of the vertical FL model, $\{\boldsymbol{v}^t\}_{t=1}^n$: confidence scores,
       $\alpha$: learning rate
Output: $\boldsymbol{\theta}^*_G$: parameters of the generator model
$\boldsymbol{\theta}_G \leftarrow \mathcal{N}(0,1)$ //initialize generator model parameters
for each epoch do
    for each batch do
        $loss = 0$
        $B \leftarrow$ randomly select a batch of samples
        $R \leftarrow \mathcal{N}(0,1)$ //initialize batch random values
        for $t \in \{1, \cdots, |B|\}$ do
            $\hat{\boldsymbol{x}}^t_{\text{target}} \leftarrow f_G(\boldsymbol{x}^t_{\text{adv}} \cup \boldsymbol{r}^t; \boldsymbol{\theta}_G)$
            $\hat{\boldsymbol{v}}^t \leftarrow f(\boldsymbol{x}^t_{\text{adv}} \cup \hat{\boldsymbol{x}}^t_{\text{target}}; \boldsymbol{\theta})$
            $loss$ += $\ell(\hat{\boldsymbol{v}}^t, \boldsymbol{v}^t)$
        $\boldsymbol{\theta}_G \leftarrow \boldsymbol{\theta}_G - \alpha \cdot \bigtriangledown_{\boldsymbol{\theta}_G} loss$ //update parameters
return $\boldsymbol{\theta}_G$
```

the path restriction attack does not apply to the RF model. Therefore, we desire to apply the GRN attack method on the RF model. However, since the objective function of the RF model is not differentiable, it is impossible to back-propagate the prediction loss through the RF model to the generator.

To address this issue, we add an additional step in the attack method. After obtaining the vertical FL model (i.e., RF), the adversary can train another differentiable model (e.g., NN) to approximate the RF model [28]. Specifically, the adversary first generates a number of dummy samples, say D_{dummy} from the whole data space, then predicts each dummy sample by the RF model. Let V_{dummy} be the prediction outputs. After that, the adversary could train an NN model $\boldsymbol{\theta}_A$ based on $(D_{\text{dummy}}, V_{\text{dummy}})$. Essentially, the NN model $\boldsymbol{\theta}_A$ is used to simulate the behavior of the RF model. As a consequence, the adversary can replace the RF model $\boldsymbol{\theta}$ with the new NN model $\boldsymbol{\theta}_A$ in Algorithm 2 to train the generator model.

VI. Experimental Evaluation

This section shows the experimental evaluation of the proposed attacks. Section VI-A presents the experimental setup. The evaluations of the attacks based on individual model prediction and multiple model predictions are described in Section VI-B and VI-C, respectively.

A. Experimental Setup

We implement the proposed attack algorithms in Python and conduct experiments on machines that are equipped with Intel (R) Xeon (R) CPU E5-1650 v3 @ 3.50GHz×12 and 32GB RAM, running Ubuntu 16.04 LTS. Specifically, we adopt *PyTorch*[2] for training logistic regression (LR) and neural networks (NN) models, and *sklearn*[3] for training decision tree (DT) and random forest (RF) models.

Datasets. We evaluate the attack performance on four real-world datasets: (i) bank marketing dataset [29], which consists of 45211 samples with 20 features and 2 classes; (ii) credit card dataset [30], which consists of 30000 samples with 23 features and 2 classes; (iii) drive diagnosis dataset [31], which consists of 58509 samples with 48 features and 11 classes. (iv) news popularity dataset [32], which consists of 39797 samples with 59 features and 5 classes. Besides, we generate two synthetic datasets with the *sklearn* library for evaluating the impact of the number of samples n in the prediction dataset on the performance of generative regression network attack. The first synthetic dataset includes 100000 samples with 25 features and 10 classes, and the second includes 100000 samples with 50 features and 5 classes. We normalize the ranges of all feature values in each dataset into $(0, 1)$.

Models. We generate the vertical FL models using centralized training and give the trained models to the adversary. This is reasonable because we consider the case that no intermediate information is disclosed during the training process, and only the final model is released. We evaluate the attack performance on the four models discussed in this paper. By default, the maximum tree depth of the DT model is set to 5. For the RF model, the number of trees and the maximum tree depth are set to 100 and 3, respectively. Besides, the NN model is composed of an input layer (with d neurons), an output layer (with c neurons), and three hidden layers (with 600, 300, 100 neurons, respectively).

Metrics. We evaluate the attack performance with two metrics. Since the equality solving attack (ESA) and generative regression network attack (GRNA) are regression tasks, we use the mean square error (MSE) per feature to measure their overall accuracy in reconstructing multiple target features. Specifically, the MSE per feature is calculated as:

$$\text{MSE} = \frac{1}{n * d_{\text{target}}} \sum\nolimits_{t=1}^{n} \sum\nolimits_{i=1}^{d_{\text{target}}} (\hat{x}^t_{\text{target},i} - x^t_{\text{target},i})^2 \qquad (9)$$

where n is the number of samples in the prediction dataset, d_{target} is the number of target features, $\hat{\boldsymbol{x}}^t_{\text{target}}$ and $\boldsymbol{x}^t_{\text{target}}$ are the inferred feature values and the ground-truth of the t-th sample, respectively. For the path restriction attack (PRA), we measure the correct branching rate (CBR). Specifically, we first randomly select a path from all the possible prediction paths computed by PRA, then accordingly measure the fraction of inferred feature values that belong to the same branches as those computed by the ground-truth. For each experiment, we conduct 10 independent trials and report the average result.

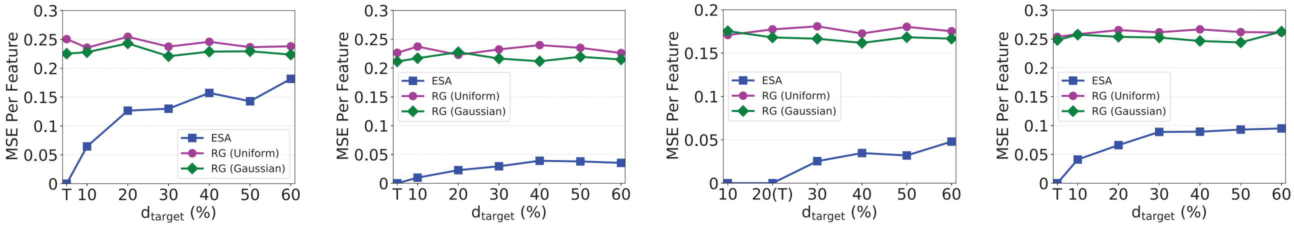
Baselines. For ESA and GRNA, we use two baselines that randomly generate samples from $(0, 1)$ according to a Uniform distribution $\boldsymbol{U}(0, 1)$ and a Gaussian distribution $\boldsymbol{N}(0.5, 0.25^2)$. This Gaussian distribution can ensure that at least 95% samples are within $(0, 1)$. For PRA on the DT model, we adopt a baseline that randomly selects a prediction path and evaluates CBR along that path. Both baselines are called *random guess* in the following presentation.

B. Evaluation of Attacks Based on Individual Prediction

For ESA and PRA based on individual model prediction (see Section IV), we evaluate the attack performance *w.r.t.* the number of features d_{target} owned by the attack target P_{target}. In

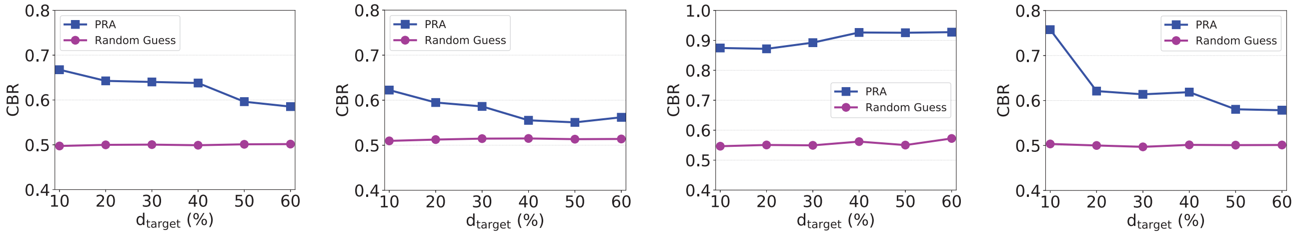
[2] https://pytorch.org/

[3] https://scikit-learn.org/stable/



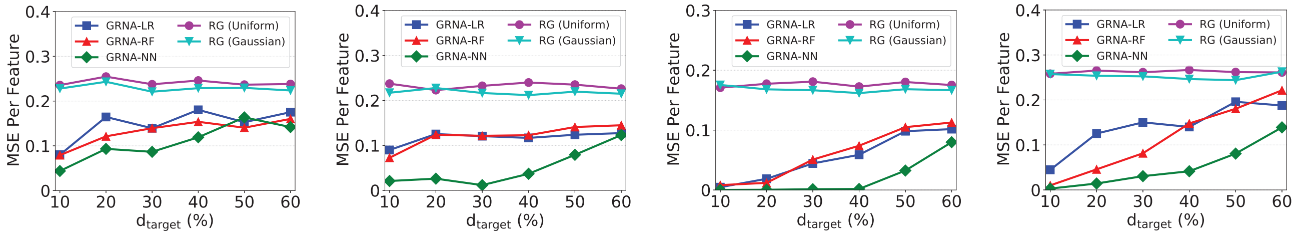
(a) Bank marketing (b) Credit card (c) Drive diagnosis (d) News popularity

Fig. 4: Evaluation of equality solving attack *w.r.t.* MSE per feature



(a) Bank marketing (b) Credit card (c) Drive diagnosis (d) News popularity

Fig. 5: Evaluation of path restriction attack *w.r.t.* CBR



(a) Bank marketing (b) Credit card (c) Drive diagnosis (d) News popularity

Fig. 6: Evaluation of generative regression network attack *w.r.t.* MSE per feature

particular, we vary the fraction of d_{target} (i.e., over the number of total features d) by $\{10\%, 20\%, 30\%, 40\%, 50\%, 60\%\}$.

Effect of d_{target} on ESA. Fig. 4 shows the attack performance of ESA *w.r.t.* MSE per feature. For all the datasets, if the threshold condition $d_{\text{target}} \leq c - 1$ is satisfied (denoted by 'T' in each sub-figure), the MSE per feature is 0. This is in line with what we have discussed in Section IV-A. For example, there are 11 classes in the drive diagnosis dataset; thus in Fig. 4c, the unknown feature values can be precisely inferred when $d_{\text{target}} = 10$ (i.e., 20% in percentage).

Moreover, we observe that even if the threshold condition is not satisfied, ESA can still find a good inference of $\boldsymbol{x}_{\text{target}}$, which is greatly superior to the random guess methods (e.g., in Fig. 4b and 4c). Besides, the MSE per feature increases as the fraction of d_{target} increases. This is expected since the larger d_{target} is, the more unknown variables in the equations, making the estimation more biased. In particular, the MSE increase of the Bank dataset is much larger than those of other datasets. As mentioned in Section IV-A, the solution $\hat{\boldsymbol{x}}_{\text{target}}$ computed by the pseudo-inverse matrix has the minimum Euclidean norm among all solutions, i.e., $||\hat{\boldsymbol{x}}_{\text{target}}||_2 \leq ||\boldsymbol{x}_{\text{target}}||_2$. Therefore, we can obtain an upper bound for $\text{MSE}(\hat{\boldsymbol{x}}_{\text{target}}, \boldsymbol{x}_{\text{target}})$, which is $\frac{1}{d_{\text{target}}}\sum_{i=1}^{d_{\text{target}}} 2x^2_{\text{target},i}$[4]. Upon that, we compute the upper bound for Bank, Credit, Drive, and News datasets, which are 0.60, 0.14, 0.45, and 0.34, respectively. In general, the larger the upper bound is, the worse the attack accuracy the adversary could achieve. This explains why the MSE of Bank increases faster than others. Besides, ESA achieves better results on Drive than on News as the adversary has more equations (i.e., $c - 1 = 10$) for Drive than those for News (i.e., $c - 1 = 4$).

Effect of d_{target} on PRA. Fig. 5 shows the attack performance of PRA *w.r.t.* CBR. Notice that the comparison between PRA and the random guess methods is similar to Fig. 4, where the attack accuracy degrades as the fraction of d_{target} goes up. In general, more target features would lead to more possible prediction paths. Therefore, the probability for the adversary to select the correct path is reduced. Note that the CBR of Drive is stable or even slightly increases as d_{target} increases. There are two reasons. First, the Drive dataset has 11 classes, which is much more than those in the other datasets (2 in Bank and Credit, 5 in News). As such, in Drive, each class corresponds to a smaller number of tree paths. Therefore, given one specific class, there exist only a small number of candidate prediction paths, which leads to an improved CBR. In this

[4] Due to the space limitation, we defer this proof to our arXiv paper.

case, the adversary would not gain a significant advantage by knowing more features. Second, the decision tree model only selects informative features during training, which means the increase of d_{target} in Drive does not necessarily increase the number of unknown features in the tree since some features may never be selected. As a consequence, a larger d_{target} do not always decrease the CBR of the PRA attack.

C. Evaluation of Attacks Based on Multiple Predictions

The generator model of GRNA is a multilayer perceptron with an input layer with size d, an output layer with size d_{target}, and three hidden layers (with 600, 200, 100 neurons, respectively). Besides, we employ Layer Normalization [33] after each hidden layer to stabilize the hidden states. For the model that imitates the RF model (see Section V-B), inspired by [28], we apply another multilayer perceptron with two hidden layers (with 2000 and 200 neurons, respectively).

Effect of d_{target} on GRNA. Fig. 6 shows the attack performance of GRNA for three models (i.e., LR, RF, and NN) on four real-world datasets *w.r.t.* MSE per feature. Similarly, we vary the fraction of d_{target}. The trends of the three models on all the datasets are similar, i.e., the MSE per feature all goes up as the fraction of d_{target} increases. This is because GRNA relies on the feature correlations between $\boldsymbol{x}_{\text{adv}}$ and $\boldsymbol{x}_{\text{target}}$ to infer the unknown feature values. The learned correlations would become weaker if the fraction of d_{target} is larger, leading to relatively worse attack performance. However, even when the fraction of d_{target} is 60%, the GRNA still achieves a much better inference than the random guess methods, demonstrating its effectiveness. In addition, GRNA with the NN model performs better than that with the LR and RF models. The reason is that the NN model has more complicated decision boundaries than the other two models, thus greatly limiting the possible distributions of $\boldsymbol{x}_{\text{target}}$ given the same $\boldsymbol{x}_{\text{adv}}$ and $\boldsymbol{v}$. Meanwhile, with more parameters, the NN model itself can capture more important information about the feature correlations than others, resulting in better attack performance.

As the NN model imitating the RF model only approximates the feature thresholds on the internal tree nodes, we also apply the CBR metric for evaluating GRNA on the RF model. Fig. 7 illustrates the performance for varying the fraction of d_{target}. Results demonstrate that GRNA recovers many more branches in the trees than the random guess methods. For example, if the fraction of d_{target} is 10%, GRNA correctly infers more than 80% of the tree branches *w.r.t.* the bank marketing and drive diagnosis datasets given the generated feature values.

Effect of n in the prediction dataset on GRNA. As the generator model is trained on multiple predictions, we investigate the effect of the number of predictions n using two synthetic datasets and two real-world datasets. For a dataset $\mathcal{D}$, we first use half of the dataset for model training and testing, then randomly select $n = \{10\%, 30\%, 50\%\} \times |\mathcal{D}|$ samples from the remaining part as the prediction dataset to train the generator using GRNA. Fig. 8 shows the attack performance *w.r.t.* MSE per feature. Overall speaking, the trends on the four datasets demonstrate that the more samples in the prediction dataset, the less MSE per feature the adversary can obtain.

In other words, the adversary could accumulate more prediction outputs in the long term to improve his attack accuracy.

Effect of data correlations on GRNA. Notice that the performances of the LR and RF models are lower than that of the NN model (see Fig. 6). The reason is that a small part of the inferred feature values is far from the ground-truth, leading to a relatively low overall attack performance. To further explore this phenomenon, we analyze the impact of data correlations between each target feature in $\boldsymbol{x}_{\text{target}}$ and the adversary's features $\boldsymbol{x}_{\text{adv}}$ as well as the prediction output $\boldsymbol{v}$. Specifically, the data correlation is defined as

$$corr(\boldsymbol{x}_{\text{adv}}, \boldsymbol{x}_{\text{target},i}) = \frac{1}{d_{\text{adv}}} \sum_{j=1}^{d_{\text{adv}}} abs(r(\boldsymbol{x}_{\text{adv},j}, \boldsymbol{x}_{\text{target},i})), \quad (10)$$

$$corr(\boldsymbol{v}, \boldsymbol{x}_{\text{target},i}) = \frac{1}{c} \sum_{j=1}^{c} abs(r(v_j, \boldsymbol{x}_{\text{target},i})), \quad (11)$$

where $r(a, b)$ is the Pearson correlation coefficient between a and b, and $abs(\cdot)$ is the absolute coefficient. The absolute value is adopted to focus on the magnitude of correlations. Essentially, the larger the two coefficients are, the easier the adversary can learn the feature correlations via GRNA.

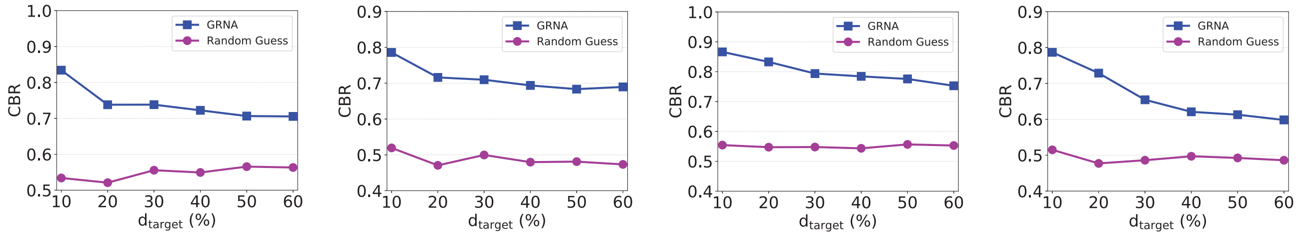
Fig. 9 shows the correlations computed by Eqn (10) and (11). The fractions of d_{target} used for the bank marketing (in Fig. 9a) and credit card (in Fig. 9b) datasets are 40% and 30%, respectively. In addition, the x-axis represents the MSE for each feature in $\boldsymbol{x}_{\text{target}}$. We can observe that the correlations *w.r.t.* both $\boldsymbol{x}_{\text{adv}}$ and $\boldsymbol{v}$ impact the attack performance of GRNA. A weaker correlation between $\boldsymbol{x}_{\text{target},i}$ with $\boldsymbol{x}_{\text{adv}}$ and $\boldsymbol{v}$ results in a lower inference accuracy, such as features 1 and 3 in Fig. 9a and features 4 and 6 in Fig. 9b. The rationale is that a weak correlation implies that the change of the unknown feature value only has a minor impact on the generated sample and the prediction output; thus, its value range is broader than the other unknown features, leading to an inaccurate inference.

Another insight is that GRNA can achieve different reconstruction accuracy regarding different features. The MSE metric computed by Eqn (9), which averages the reconstruction errors on all target features, is mainly used to evaluate the overall performance of our attacks. In real-world applications, we should note that features closely correlated to $\boldsymbol{x}_{\text{adv}}$ can be more precisely reconstructed than the relatively independent ones. For example, after obtaining the inferred deposit and shopping information based on a user's income, the adversary may be more convinced of the inferred deposit as it is more relevant to the income feature.

VII. Countermeasures

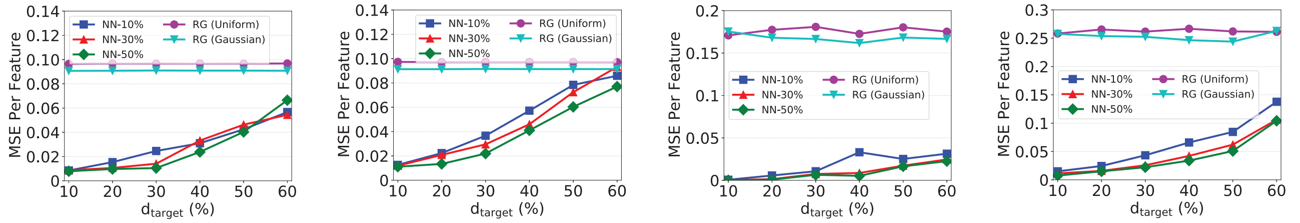
In this section, we discuss several potential defense methods that may mitigate the proposed feature inference attacks.

Rounding confidence scores. In ESA, the adversary relies on the exact linear equations for the inference. Thus, a possible defense to ESA is to coarsen the confidence scores $\boldsymbol{v}$ returned



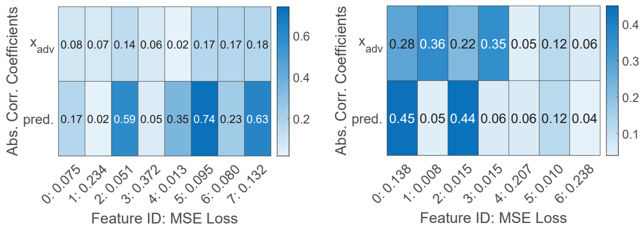
(a) Bank marketing (b) Credit card (c) Drive diagnosis (d) News popularity

Fig. 7: Evaluation of generative regression network attack on the RF model *w.r.t.* CBR



(a) Synthetic dataset 1 (b) Synthetic dataset 2 (c) Drive diagnosis (d) News popularity

Fig. 8: Effect of number of predictions *w.r.t.* MSE per feature



(a) Bank marketing (LR model) (b) Credit card (RF model)

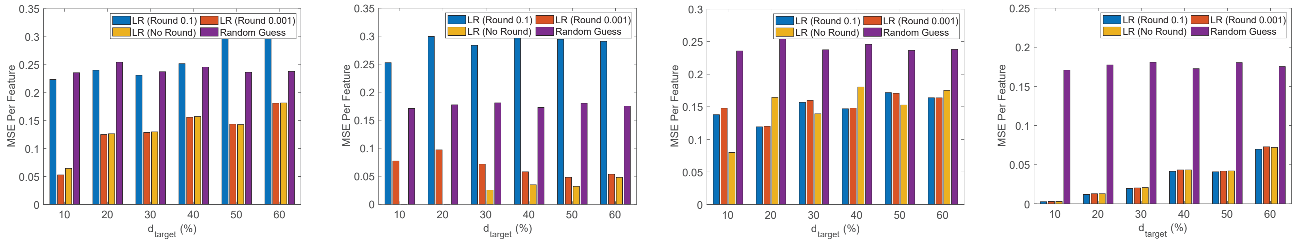
Fig. 9: Effect of data correlations between each feature in $\boldsymbol{x}_{\text{target}}$ with $\boldsymbol{x}_{\text{adv}}$ and $\boldsymbol{v}$

to the active party, for example, round $\boldsymbol{v}$ down to b floating point digits before revealing it. Since the performances of the two random guess methods are similar, we only include the random guess with uniform distribution in this set of experiments. Fig. 10a-10b show the effect of this strategy on two datasets, respectively. We can observe that when rounding to 0.1 (i.e., $b = 1$), the MSE per feature is higher than the random guess method, and the result is relatively stochastic. This is because that the equation result in Eqn (3) or (7) is related to $\ln \boldsymbol{v}$, a change of $\boldsymbol{v}$ with respect to 0.1 has thus a big impact on that result, leading to an inaccurate inference. In contrast, rounding to 0.001 (i.e., $b = 3$) only has a small impact because the result would be mainly determined by the former three floating point digits. Fig. 10c-10d show the effect of the rounding strategy in GRNA for the LR model. The results illustrate that GRNA is insensitive to the rounding of confidence scores, and the adversary can obtain a similar performance comparing to that without rounding. The reason is that GRNA learns the overall correlation between the adversary's features and the attack target's features, and the low-precision prediction outputs still indicate this pattern.

Dropout for neural networks model. Overfitting is considered to be an important factor in several inference attacks in the training stage [15, 34], as the trained model may memorize some underlying information of the data. We utilize a state-of-the-art regularization technique called dropout [35] for training the vertical NN model to avoid overfitting. As depicted in Fig. 11a-11b, using dropout slightly leads to a higher MSE per feature (i.e., degrade the attack performance), because dropout encourages NN models to memorize less distribution information of underlying datasets. Nevertheless, the adversary can still have a good inference because the generator model learns the overall feature correlations.

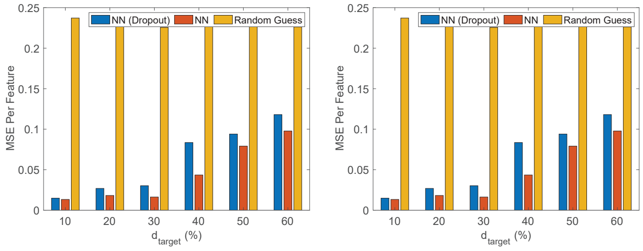
Pre-processing before collaboration. Based on the experimental results in Section VI, we observe that: (i) if the number of classes is relatively large, ESA can precisely recover the attack target's feature values, and PRA can correctly infer most branches in the tree model; (ii) if the adversary's features are highly correlated with the attack target's features, the attack performance of GRNA could be greatly improved. These observations inspire the parties to execute a pre-processing step that mitigates the potential privacy risks before data collaboration. First, the parties check the relationship between the number of classes and the number of their contributed features, ensuring no obvious privacy vulnerabilities. Second, the parties collaboratively execute a secure protocol (e.g., using secure multiparty computation [4]) to compute their feature correlations and remove those closely related features.

Post-processing for verification. The parties can also execute an additional verification step to check if the prediction output would lead to privacy leakage before revealing to the active party. For example, if the parties utilize secure hardware to compute model predictions, then they can mimic these attacks inside the secure enclaves. Specifically, if the possible leakage exceeds a pre-defined threshold for any party, they do not reveal this prediction for privacy protection. In other words, the prediction output is released only when it satisfies the privacy of all parties. However, this post-processing step may incur huge overheads to the computation of model predictions.



(a) ESA on bank marketing (b) ESA on drive diagnosis (c) GRNA on bank marketing (d) GRNA on drive diagnosis

Fig. 10: Effect of the rounding strategy: (a)-(b) and (c)-(d) illustrate ESA and GRNA *w.r.t.* LR, respectively.



(a) GRNA on credit card (b) GRNA on news popularity

Fig. 11: Effect of the dropout strategy in GRNA *w.r.t.* NN

Hide the vertical FL model. Another consideration is to hide the trained vertical FL model using ciphertexts or secure multiparty computation compatible values, such that the adversary does not have access to the plaintext model in the attack. This method could mitigate these attacks, but the active party cannot justify if the trained model is reasonable or not and cannot have a good interpretation of the prediction output for making important decisions. A possible alternative is to use explainable predictions instead of revealing the trained vertical FL model, such that the contribution of each feature is provided to the active party for justification. However, it is still an open problem whether those explainable predictions would cause new privacy leakages.

Differential Privacy (DP). DP [36] is a state-of-the-art privacy protection tool. However, it is unsuitable in our setting. In particular, DP requires that any information from sensitive data should be released using a randomized function, such that even if we arbitrarily change a record in the function's input, the distribution of the function's output remains roughly the same. In our context, the sensitive data is an unlabelled record, and the function that releases information is a machine learning model that predicts the record's label. If we are to achieve DP, then we should inject noise into the model, such that even if we arbitrarily modifies the input record, the distribution of the model's output label is almost unchanged. In other words, the model should provide roughly the same prediction for any input record. This apparently renders the model useless. Therefore, DP is unsuitable for our problem.

VIII. Related Work

Vertical federated learning. With horizontal FL being thoroughly studied [7], vertical FL is receiving increased attention recently [4, 8, 10, 11, 26, 37]. [10, 37] apply partially homomorphic encryption [13] in the LR and GBDT models to protect the information exchanged among the parties. [26] and [4] utilize secure multiparty computation to provide strong data security for the NN and tree-based models, respectively. Also, secure hardware could protect the parties' private data during the training and prediction stages using secure enclaves [11]. By allowing that the active party's labels can be shared with others, [8] proposes to aggregate the local predictions computed by each party into a final prediction.

Nevertheless, existing vertical FL solutions focus on protecting data privacy during the training or prediction process. In this paper, we consider that the computation of model training and model prediction is secure enough, and the adversary only utilizes the computation output (i.e., the trained model and model predictions) to infer the target feature values.

Inference attacks on federated learning. Recent studies have demonstrated that FL is vulnerable to multiple types of inference attacks, such as *membership inference* [15, 16], *property inference* [16], and *feature inference* [17, 18]. Membership inference [15, 16] aims to determine whether a specific record is in a party's training dataset or not, and the attacking effectiveness is closely related to the overfitting nature of ML algorithms [34, 38]. However, this attack does not exist in vertical FL as every party knows all the training sample ids intrinsically. Property inference [16] attempts to extract some underlying properties or statistics of a party's training dataset, which are uncorrelated to the training task [16, 39]. Feature inference [17, 18] is to recover the samples used in a party's training dataset. Unfortunately, these inference attacks are only applicable to the training stage of horizontal FL, while we focus on the feature inference in the prediction stage of vertical FL, which is more challenging because the federated model is not aware of the predicting samples beforehand. To our knowledge, this is the first work that addresses this problem.

Other related studies. In addition to feature inference in FL, another line of research tries to infer private features based on a centralized ML model. For example, [19] proposes an attack to infer patients' genetic markers given black-box access to a linear regression model. Nevertheless, the adversary needs background information of all features except the targeting feature, making this attack relatively impractical. Similarly, the inference attack to the DT model in [20] requires a marginal prior distribution for each feature, and the image reconstruction attack also relies on auxiliary information to define the cost function of the attack. In contrast, our attack methods do not rely on any background information on the attack target's data distribution.

The basic idea of [22] is similar in spirit to ours, i.e., inferring sensitive features (pixels) based on known image patches and white-box models. But this scheme requires pre-training on public images for obtaining the general image distributions, whereas our approach does not rely on the prior data distribution of the target features. Besides, [22] aims to recover the training samples, whereas our attack methods mainly focus on reconstructing the predicting samples.

IX. Conclusion

We present two specific attacks based on individual prediction output: equality solving and path restriction, for logistic regression and decision tree models, respectively. Furthermore, we design a general attack based on multiple prediction outputs for neural networks and random forest models. To our knowledge, this is the first work that investigates the privacy leakages in vertical FL. The experimental results demonstrate the accuracy of the proposed attacks and highlight the need for designing private algorithms to protect the prediction outputs.

Acknowledgement

This research is supported by Singapore Ministry of Education Academic Research Fund Tier 3 under MOEs official grant number MOE2017-T3-1-007.

References

[1] "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). oj, 2016-04-27."

[2] "California consumer privacy act. bill no. 375 privacy: personal information: businesses. https://leginfo.legislature.ca.gov/. 2018-06-28."

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

[4] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 2090–2103, 2020.

[5] 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 *CCS*, 2017, pp. 1175–1191.

[6] W. Zheng, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Helen: Maliciously secure coopetitive learning for linear models," in *IEEE S&P*, 2019, pp. 915–929.

[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM TIST*, vol. 10, no. 2, pp. 12:1–12:19, 2019.

[8] Y. Hu, D. Niu, J. Yang, and S. Zhou, "FDML: A collaborative machine learning framework for distributed features," in *SIGKDD*, 2019, pp. 2232–2240.

[9] J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson, "Privacy-preserving decision trees over vertically partitioned data," *TKDD*, vol. 2, no. 3, pp. 14:1–14:27, 2008.

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

[11] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, "Oblivious multi-party machine learning on trusted processors," in *USENIX Security Symposium*, 2016, pp. 619–636.

[12] Q. He, W. Yang, B. Chen, Y. Geng, and L. Huang, "Transnet: Training privacy-preserving neural network over transformed layer," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 1849–1862, 2020.

[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999, pp. 223–238.

[14] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *CRYPTO*, 2012, pp. 643–662.

[15] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Stand-alone and federated learning under passive and active white-box inference attacks," in *IEEE S&P*, 2019, pp. 1021–1035.

[16] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *IEEE S&P*, 2019, pp. 691–706.

[17] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *CCS*, 2017, pp. 603–618.

[18] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *NeurIPS*, 2019, pp. 14 747–14 756.

[19] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *USENIX Security Symposium*, 2014, pp. 17–32.

[20] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *CCS*, 2015, pp. 1322–1333.

[21] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *CSF*, 2018, pp. 268–282.

[22] 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 *CVPR*, 2020, pp. 253–261.

[23] D. J. Wu, J. Zimmerman, J. Planul, and J. C. Mitchell, "Privacy-preserving shortest path computation," in *NDSS*, 2016.

[24] S. R. H. Ling, Z. Yehong, C. M. Choon, and L. B. K. Hsiang, "Collaborative machine learning with incentive-aware model rewards," *ICML*, 2020.

[25] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *CCS*, 2017, pp. 1243–1255.

[26] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *IEEE S&P*, 2017, pp. 19–38.

[27] "Moore–penrose inverse," https://en.wikipedia.org/wiki/Moore%E2%80%93Penrose_inverse, 2020, [Online; accessed 29-Dec-2020].

[28] G. Biau, E. Scornet, and J. Welbl, "Neural random forests," *Sankhya A*, vol. 81, no. 2, pp. 347–386, 2019.

[29] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decision Support Systems*, vol. 62, pp. 22–31, 2014.

[30] I. Yeh and C. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 2473–2480, 2009.

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

[32] K. Fernandes, P. Vinagre, and P. Cortez, "A proactive intelligent decision support system for predicting the popularity of online news," in *EPIA*, 2015, pp. 535–546.

[33] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016.

[34] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE S&P*, 2017, pp. 3–18.

[35] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[36] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," in *ICDE*, 2010, pp. 225–236.

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

[38] 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," in *NDSS*, 2019.

[39] 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 *CCS*, 2018, pp. 619–633.