Feature Engineering Framework based on Secure Multi-Party Computation in Federated Learning

Sun, Litong; Du, Runmeng; He, Daojing; Zhu, Shanshan; Wang, Rui; Chan, Sammy

DOI
10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00088

Publication date
2022

Document Version
Final published version

Published in
2021 IEEE 23rd International Conference on High Performance Computing and Communications, 7th International Conference on Data Science and Systems, 19th International Conference on Smart City and 7th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Applications, HPCC-DSS-SmartCity-DependSys 2021

Citation (APA)
Sun, L., Du, R., He, D., Zhu, S., Wang, R., & Chan, S. (2022). Feature Engineering Framework based on Secure Multi-Party Computation in Federated Learning. In *2021 IEEE 23rd International Conference on High Performance Computing and Communications, 7th International Conference on Data Science and Systems, 19th International Conference on Smart City and 7th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Applications, HPCC-DSS-SmartCity-DependSys 2021* (pp. 487-494). IEEE. https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00088

2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application

Feature Engineering Framework based on Secure Multi-Party Computation in Federated Learning

1st Litong Sun
dept. School of Software Engineering
East China Normal University
Shanghai, China

2nd Runmeng Du
dept. School of Software Engineering
East China Normal University
Shanghai, China

3rd Daojing He
dept. School of Software Engineering
East China Normal University
Shanghai, China

4th Shanshan Zhu
dept. School of Software Engineering
East China Normal University
Shanghai, China

5th Rui Wang
Technische Universiteit Delft
Delft, Netherlands

6th Sammy Chan
City University of Hong Kong
Hong Kong, China

***Abstract*—Data and features often determine the upper limit of results, so that feature engineering is an important stage of federated learning. The existing research schemes all carry out feature engineering based on publicly sharing data. One is plaintext data sharing, the other is ciphertext data sharing, but both types of sharing bring security and efficiency problems. To address these challenges, we propose a feature engineering framework based on Secure Multi-party Computation, which supports multi-party participation in feature engineering and confines feature data locally to ensure data security. Moreover, the computational efficiency of the core algorithm of the framework is also improved compared with the existing methods.**

***Keywords*—Feature Engineering; Federated Learning; Secure Multi-party Computation; Privacy Protection**

I. INTRODUCTION

Data is the foundation of all business, and many joint modeling for machine learning (ML) can achieve better effect. The vigorous development of ML is because of availability of large amounts of data [1], [2]. Despite recent technological advances that have made big data storage, processing and computation more efficient, combining data from different sources remains a major challenge [3], [4]. Competitive advantage, privacy issues, regulations, as well as the question of sovereignty and jurisdiction over data have hindered many organizations on publicly sharing data [5], [6].

Secure multi-party computation (MPC), proposed by Yao, refers to the secure computation jointly carried out by two or more participants. After the computation, each participant gets the given output without any leakage of its own input information [7]. As a result, MPC technique is used in federated learning research. At present, the whole process of federated learning includes data collection, feature engineering, federated model training and so on, among which feature engineering is the most important part of machine learning [8].

There are a lot of research about feature engineering in federated learning. In this process, the party with label data needs to assist the party with only feature data in feature preprocessing under the condition that feature data and label data are not leaked [9], [10]. In the following, the party that has only the feature data matrix and lacks the label matrix is referred to as the data provider and the party who has both the feature data matrix and the label matrix is referred to as the data application party.

For most of the existing feature engineering frameworks, the data user usually encrypts the label matrix with the public key to meet the privacy protection requirements, and then sends the ciphertext matrix to the data provider, who in turn computes binning according to class ciphertext matrix [11]. However, in large scale data collection, this approach can obviously result in significant resource depletion and performance degradation. There is also direct transfer of desensitization data for feature engineering processing [12], but this cannot protect data privacy and does not conform to legal norms.

The proposed solution to the federated engineering problem is a good response to the inadequacy of existing solutions. The main contributions of this paper are as follows:

- We propose a feature engineering framework called FEFL. Compared with other frameworks, it supports multi-party participation and confines the feature data in locally, improving the safety and efficiency of the feature engineering in federated learning.
- Based on the FEFL framework, we propose a new chi-square binning method. Compared with direct encryption of feature data, this method marks the group category of feature data after grouping, and then encrypts the group category, which greatly reduces the amount of encryption and improves the computing efficiency.
- Based on the FEFL framework, we propose a new method to compute the maximal information coefficient. We discretize the feature data through the meshing schema, record the region of the feature data. Compared with other methods that encrypt the feature data and send it directly to the data application, this method does not send the actual data and has stronger security.

The rest of this paper is organized as follows. Section II discusses some related works. Section III provides some background knowledges. Section IV introduces the relevant algorithms in the framework FEFL. Section V provides experimental results of FEFL. Section VI concludes this paper.

II. RELATED WORK

Since data is the foundation of everything, feature engineering for federated learning has been studied in recent years. Fang *et al.* [11] have introduced a framework to conduct privacy-preserving and communication-preserving multi-party feature transformations. But the data from the data provider is sent to the server and the data application, and the data can be easily leaked. Li *et al.* [8] have introduced an efficient feature scoring protocol. However, in their scheme, the feature data and label matrix are encrypted by secret sharing and sent to the data application, so the security of the data cannot be guaranteed when the data is sent out locally, and the efficiency is very low when the score is computed under ciphertext. Sei *et al.* [13] have introduced a novel anonymization method (RandChiDist), which anonymizes Chi-square value testing for small samples relying mainly on the differential privacy. The data processed in the feature engineering stage is complex and irregular, and the differential privacy has too many restrictions on the input data, and a large amount of randomization is added into the output results, resulting in a sharp decline in the availability of data. Sometimes the randomization results almost cover the real results. Zhang and Chen [14] proposed the Chi-square binning method, which encrypts the label data and sends it to other participants. The label data will leave the local environment and may be leaked. In addition, a large number of encryptions leads to low computing efficiency. The authors mainly solved the privacy protection problem by desensitizing the data, but at the same time data loss is caused and the accuracy is sacrificed.

III. PRELIMINARIES

A. Security models

*1) **Ideal model:*** Assuming that there are participants $P_i, i \in [0, m-1]$ have a privacy data $(x_1,\cdots,x_n)$ respectively. They securely compute $f(x_1,\cdots,x_n) = (f_1(x_1,\cdots,x_n),\cdots,f_n(x_1,\cdots,x_n))$ without compromising the privacy data. The simplest and safest approach is to adopt the secure multi-party computation protocol under the ideal model. The ideal model assumes that there is an absolutely credible third party who will not disclose anything that should not be disclosed under any circumstances. In this assumption, the process of secure multi-party computation is as follows. Each party sends its private data $(x_1,\cdots,x_n)$ to a trusted third party, who computes $f(x_1,\cdots,x_n)$, and sends $f_i(x_1,\cdots,x_n)$ to P_i.

*2) **Semi-honest model:*** Semi-honest model is an important secure multi-party computation model in which every participant behavior is consistent with the requirements of the protocol. However, it will retain information about the computation process and try to use this information to obtain more private information about other participants after the execution of the protocol.

*3) **Threat model:*** The primary issue is that recording and storing raw, sensitive data can expose unwanted insights about a specific individual or party [15]. Considering that there will be a lot of sensitive data communication in the federated learning multi-party joint computing process, we need to explore all possibilities before designing any algorithm for attack mitigation. In this paper, we keep the private data of each party locally to resist the threat model attack.

B. Paillier cryptosystem

Paillier cryptosystem is a probabilistic cryptosystem and the specific process is as follows [16].

Key generation First select two prime numbers p, q, where $n = p{\times}q, \lambda = \text{lcm}(p-1, q-1)$ is least common multiple of p–1 and $q-1$. Select at random $g \in Z_n^*$ that makes gcd$(L(g^\lambda \bmod n^2), n) = 1$, where $L(x) = \dfrac{x-1}{N}$. The public key is (g, n) and the private key is λ.

Encryption For plaintext messages m, select a random number $r, r < n$ and compute

$$Enc(m) = g^m r^n \bmod n^2 \tag{1}$$

Decryption For ciphertext $Enc(m)$, compute

$$Dec(m) = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n^2 \tag{2}$$

Homomorphism Direct verification shows that the Paillier cryptosystem has additive homomorphism.

$$\begin{aligned} Enc(m_1)Enc(m_2) &= g^{m_1} r_1^N g^{m_2} r_2^n \bmod n^2 \\ &= g^{m_1+m_2}(r_1 r_2)^N \bmod n^2 \\ &= Enc(m_1 + m_2) \end{aligned} \tag{3}$$

C. Feature Engineering in Federated Learning

Feature selection is an important step in feature engineering, and Chi-square binning is often carried out in the process of feature selection [17]. In order to improve the modeling effect, federated learning needs to evaluate the correlation between variables by Maximal Information Coefficient in the feature engineering stage [18].

*1) **Chi-square binning:*** There is a bottom-up method of binning in supervised learning called Chi-square binning that relies on the Chi-square test. The Chi-square test is defined as the merging of contiguous intervals with the smallest Chi-square value until a deterministic stopping criterion is met. Thus, two adjacent intervals can be merged if they have very similar class distributions, otherwise, they should remain separate [19], [20].

*2) **Maximal Information Coefficient:*** The maximal information coefficient (MIC) is used to measure the linear or nonlinear strength of two variables X and Y. MIC can not only find the linear functional relationship between variables, but also can find the nonlinear functional relationship [21].

IV. Feature Engineering Framework in Federated Learning

In this section, we present the framework of Feature Engineering for Federated Learning (FEFL), which involves the optimization and implementation steps of several important algorithms.

A. The framework of FEFL

FEFL enables participants to do feature engineering, without having to share feature data publicly or encrypted. Compared to other frameworks, it provides a more secure data processing environment for data owners. The framework focuses on the improvement of data privacy protection and computation overhead, and the improvement is very obvious. The framework entity relationship is shown in Figure 1 below:

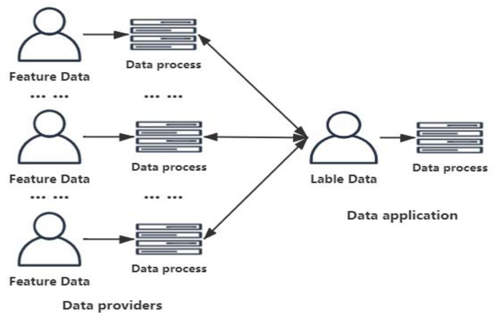


Figure 1. Framework entity relationship

As shown in Figure 1, we set the participants information and feature data information in advance. There are data providers $P_i, i \in [0, m-1]$, holding feature data, where $F_i = \{X_0, \cdots, X_z\}, k \in [0, z-1]$ is used to represent the feature set of feature data $X_k = \{x_0, \cdots, x_{n-1}\}, X_k \in M^{n\times 1}$. Here, we require that feature set F_i has no intersection, namely $F_p \cap F_q = \varnothing, p \neq q \in [0, m-1]$. The data application P_0 holds the label data $Y = \{y_0, \cdots, y_{n-1}\} \in M^{n\times 1}$, where $id = \{0, 1, 2, \ldots, n-1\} \in M^{n\times 1}$ is used to represent the row index of feature data F_i and lable data Y. In this paper, all participants have implemented sample alignment before algorithms executed.

B. Chi-square binning

In [14], the data application party encrypts all the label data Y and sends it to the data provider. The problem of [14] is that encrypting all label data Y will consume a lot of resources and reduce the computing efficiency. In addition, the data application party send the label data Y to the data provider, which may expose the label data.

Compared with [14], Algorithm 1 to be presented below is better. First, the data provider groups the feature data X, mixes an appropriate amount of false groups, uses 0,1 markers respectively to distinguish the true group from the false group, and then encrypts the group markers. With the same amount of data, the number of grouping markers is far less than the number of lable data Y, and the number of encryption is as little as half of that in [14], which greatly improves the computing efficiency. Secondly, the data application in [14] encrypts the label data and sends it to the data provider, and the label data leaves the local environment. In Algorithm 1, the label data and feature data do not leave the local environment, which is more secure.

1) ***Algorithm design****:* The specific steps of federated chi-square binning algorithm combined with homomorphic encryption scheme are referred to as Algorithm 1, which is executed by the data prividers. Algorithm 2 is a child of Algorithm 1, which is executed by the data application.

Algorithm 1 requires the data prividers P_i execution

Input: P_i holds $F_i = \{X_0, \cdots, X_z\}, k \in [0, z-1]$, $X_k = \{x_0, \cdots, x_{n-1}\}$, $id = \{0, 1, 2, \cdots, n-1\}$.
Output: Chi-square binning result $Bins_{F_i}, i \in [1, m-1]$

```
1: for $i = 1$ to $m - 1$ do
2:     Initializes the grouping matrix $Group_{F_i} = \varnothing \in M^{0\times 0}$
3:     for $k = 0$ to $z - 1$ do
4:         $t_k \leftarrow Grouping(X_k), t_k \in id$
5:         $v_k \leftarrow RandomGrouping(X_k), v_k \in id$
6:         $Group_{t_k}(t_k, Enc_{t_k}) \leftarrow$
           $HorizontalConnect(t_k, Enc(1))$
7:         $Group_{v_k}(v_k, Enc_{v_k}) \leftarrow$
           $HorizontalConnect(v_k, Enc(0))$
8:         $Group_{X_k}(x_k, Enc_{x_k}) \leftarrow$
           $VerticalConnect(Group_{t_k}, Group_{v_k})$
9:     end for
10:    $Group_{F_i} \leftarrow$
       $Disrupt(HorizontalConnect(Group_{X_{k-1}}, Group_{X_k}))$
11:    $P_i$ sends $Group_{F_i}$ to $P_0$
12: end for
13: $Inf_{F_i} \leftarrow Algorithm2$
14: for $i = 1$ to $m - 1$ do
15:     for $k = 0$ to $z - 1$ do
16:         $Bins_{t_k} \leftarrow Inf_{X_k} \leftarrow Inf_{F_i}$
17:     end for
18:     $Bins_{F_i} \leftarrow HorizontalConnect(Bins_{t_{k-1}}, Bins_{t_k})$
19: end for
```

First, the data provider P_i needs to group the feature data F_i by the steps 1-11 of Algorithm 1. P_i initializes a grouping matrix $Group_{F_i} \in M^{0\times 0}$. P_i groups feature data X_k by category through function $Grouping()$ to get the true group t_k. The t_k is obtained by putting the row index of X_k of the same category into an array. P_i uses paillier encryption (1) to encrypt the group tag 1,0 to obtain $Enc_{t_k} = Enc(1)$,$Enc_{v_k} = Enc(0)$. P_i uses the horizontal connection function $HorizontalConnect()$ to connect t_k and Enc_{t_k} to get the true group information of feature data $Group_{t_k}(t_k, Enc_{t_k})$.

The data provider P_i uses the random grouping function $RandomGrouping()$ to get random indexes from array id to obtain false group v_k. P_i uses the function $HorizontalConnect()$ to connect v_k and $Enc(v_k)$ to get the false group information $Group_{v_k}(v_k, Enc_{v_k})$. Then, P_i verticaly connects the $Group_{t_k}, Group_{v_k}$ by the function $VerticalConnect()$ to obtain the $Group_{X_k}$. P_i horizontaly connects $Group_{X_{k-1}}, Group_{X_k}$ by the function $HorizontalConnect()$ and uses the funcation $Disrupt()$ to scramble the order of groups to obtain the grouping information $Group_{F_i}$. P_i sends $Group_{F_i}$ to data application party P_0. The step 13 of Algorithm 1 requires the data application party execute the Algorithm 2.

Algorithm 2 requires the data application P_0 execution

Input: P_0 holds $Y = \{y_0, \cdots, y_{n-1}\}, id = \{0, \cdots, n-1\}$.
Output: The number of response and non-response after feature data grouping $Inf_{F_i}, i \in [1, m-1]$

```
1:  for $i = 0$ to $m - 1$ do
2:      Get $Group_{F_i}$ by the steps[1-12] of Algorithm 1
3:      for $k = 0$ to $z - 1$ do
4:          $Group_{X_k}(x_k, Enc_{x_k}) \leftarrow Group_{F_i}$
5:          $Y_{x_k} \leftarrow Map(x_k, Enc_{x_k}, Y)$
6:          $Inf_{x_k}(Group_y, Group_s, Group_n, Enc_{x_k}) \leftarrow Y_{x_k}$
7:      end for
8:      $Inf_{F_i} \leftarrow HorizontalConnect(Inf_{x_{k-1}}, Inf_{x_k})$
9:      $P_0$ sends $Inf_{F_i}$ to $P_i$
10: end for
```

According to Algorithm 2, P_0 got the grouping information $Group_{F_i}$ from P_i. Function Map is used to map the grouping information $x_k, x_k \in id$ array to the label data Y to obtain the label values Y_{x_k}, the elements of x_k is the index of lable data Y. For example, the grouping interval $x_k = [0, 2]$ corresponds to the values of the label data $Y_{x_k} = [y_0, y_2]$. The binary classification problems in machine learning, the sample with lable value of 1 represents the response sample, and the label value of 0 indicates the non-response sample. Therefore, the number of response samples $Group_y$ in this group can be calculated according to the lable value corresponding to the grouping information.

TABLE I
THE NUMBER OF RESPONSE SAMPLES

index	$Group_y(j)$	$Group_n(j+1)$	$R_i(Group_s)$
i	$sum(Y_{x_k}^{(i)})$	$Group_s^{(i)} - Group_y^{(i)}$	$len(x_k^{(i)})$
$i+1$	$sum(Y_{x_k}^{(i+1)})$	$Group_s^{(i+1)} - Group_y^{(i+1)}$	$len(x_k^{(i+1)})$
C_j	$sum(Group_y)$	$sum(Group_n)$	$sum(Group_s)$

As shown in Table I, x_k represents the set of group, assuming x_k contains s groups,then $i \in [0, s-1]$. The number of response samples in the ith group $Group_y$ is the sum of lable values corresponding to the group $Group_y^{(i)} = sum(Y_{x_k}^{(i)})$. The total samples of the ith group is the number of elements in the group $Group_s^{(i)} = len(x_k^{(i)})$, the function $len()$ finds the number of elements in an array, and the non-response samples in ith group is computed by $Group_n^{(i)} = Group_s^{(i)} - Group_y^{(i)}$. According to the Algorithm 2, P_0 gets the $Inf_{x_k}(Group_y, Group_s, Group_n)$, and horizontaly connects the Inf_{x_k} and $Inf_{x_{k-1}}$ to get the Inf_{F_i}. P_0 sends Inf_{F_i} to P_i.

P_i obtains Inf_{F_i} from P_0, according to steps 14-19 of Algorithm 1, it can compute the chi-square value χ^2, the adjacent groups with the smallest chi-square value are merged until the group limit is reached. Then P_i can obtain the result of chi-square binning $Bins_{F_i}$.

*2) **Correctness analysis of Algorithm 1:*** The correctness of Algorithm 1 and Algorithm 2 can be guaranteed by the basic principle of chi-square binning and the special coding method used in this paper. Then we give the Theorem 1 as follows:

Theorem 1. *Algorithms 1 and 2 can compute the Chi-square binning correctly.*

Proof: Algorithm 2 is a sub algorithm of Algorithm 1, and we directly prove the correctness of Algorithm 1. According to the steps 1-13 of Algorithm 1, P_i obtains the Inf_{F_i}, then can compute the chi-square binning. P_i decrypts the Enc_{x_k} to abtain the samples information of true groups Inf_{t_k}, where the $Dec(Enc(x_k)) = 1$. According to the number of response samples and the number of non-response samples in the ith true groups, the expected frequency E_{ij} can be obtained. The calculation process is as follows:

$$E_{ij} = \frac{R_i \times C_j}{N} = \frac{len(x_k^{(i)}) \times sum(Group_y)}{sum(Group_s)} \tag{4}$$

Equation (4), N is the total number of samples, R_i is the numbers of samples for the group i, C_j is the proportion of the sample of class j in the whole.

$$\begin{aligned}\chi^2 &= \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \\ &= \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{(Group_y^{(ij)} - \frac{R_i \times C_j}{N})^2}{\frac{R_i \times C_j}{N}}\end{aligned} \tag{5}$$

Equation (5), A_{ij} denotes the number of instances of class j from interval i, E_{ij} denotes the expected frequency of A_{ij}. According to the number of sample A_{ij}, expected frequency E_{ij}, P_i can calculate the Chi-square value χ^2 of adjacent groups, and the groups with the lowest Chi-square value can be combined until the group number limit is reached.

P_i uses the private key in the Paillier encryption system to decrypt the $Enc(x_k)$ to obtain the $Dec(Enc(x_k))$. The decryption function $Dec()$ is given by (2) in Section III. P_i can filter the false group in the process of compution the chi-square value when calculating the ith group by Enc_{x_k}. P_i decrypts the Chi-square value of false group is $Dec(Enc(0)) = 0$. Thus, the real Chi-square binning $Bins_{F_i}$ is obtained. ■

*3) **Security analysis of Algorithm 1:*** During the binning process,the data provider P_i mixes the group information into the false group with the same amount as the real group.

P_i marks the real group as $Enc(1)$ and the false group as $Enc(0)$. Then, P_i verticaly connects the real group with the false group, disrupts the order and sends to the data application P_0. Honest but curious participants hope to infer the feature data from the grouping information. However, in Algorithm 1, these participants do not have the private key of P_0, and the grouping order is also disrupted, so as to avoid the disclosure of private information. In addition, we construct an emulator, and have the following theorem:

Theorem 2. *Algorithms 1 and 2 can safely compute the chi-square binning in federated learning.*

Proof: Algorithm 2 is a sub algorithm of Algorithm 1. We use the simulation example to prove the Theorem 2 strictly. We first prove the security of the P_0 label data. Since the participants $P_i, i \in [1, m-1]$ are all equal, we only need to prove that Algorithm 1 is safe for the set of conspirators $I = \{P_1, \cdots, P_{m-1}\}$. For the set of conspirators composed of $m-1$ participants in I, the corresponding simulator S needs to be constructed. S first applies the Paillier public key system, and the public key is set as pk, and the corresponding private key is sk. The emulator S runs as follows:

(i). Simulator S receives the input $(I, F_I, f_I(F_0, \cdots, F_{m-1}))$, randomly selects $F_0' = (f_0', \cdots, f_k')$, such that $f_I(F_0, \cdots, F_{m-1}) = f_I(F_0', \cdots, F_{m-1})), F_0' \overset{c}{\equiv} F_0$.

(ii). Simulator S performs steps 2-21 of Algorithm 1 to compute binnings $Bins' = \{Bins_{F_0'}, \cdots, Bins_{F_{m-1}}\}$.

(iii). In the execution of the Algorithm 1, $Bins = \{Bins_{F_0}, \cdots, Bins_{F_{m-1}}\}$, the conspirators I to obtain the information view: $view_I^{\pi}(F_1, \cdots, F_{m-1}, f_I(F_0, \cdots, F_{m-1})) = \{F_1, \cdots, F_{m-1}, Bins, f_I(F_0, \cdots, F_{m-1})\}$, and then let $S(F_1, \cdots, F_{m-1}, f_I(F_0, \cdots, F_{m-1})) = \{F_1, \cdots, F_{m-1}, Bins', f_I(F_0', \cdots, F_{m-1})\}$.

It can be concluded from the above formula that the conspirators I did not receive any data information of the non-conspirators during the execution of the algorithm, even encrypted messages. Finally, $\{Bins_{F_0}, \cdots, Bins_{F_{m-1}}\} = \{Bins_{F_0'}, \cdots, Bins_{F_{m-1}}\}, Bins_{F_0'} \overset{c}{\equiv} Bins_{F_0}$, therefore $\{view_I^{\pi}(F_1, \cdots, F_{m-1}, f_I(F_0, \cdots, F_{m-1})\} \overset{c}{\equiv} \{S(F_1, \cdots, F_{m-1}, f_I(F_0, \cdots, F_{m-1})\}$.

Secondly, we need to prove the security of the feature data $P_i, i \in [1, m-1]$. Since the status of participant $P_i, i \in [1, m-1]$ is all equal, we only need to prove that Algorithm 2 is safe for participant $I' = P_0$. For the participant I', the corresponding simulator S' needs to be constructed. S' first applies the Paillier public key system. Let its public key be pk', and the corresponding private key be sk'. The Simulator S' mode is similar to the steps in the above Simulator operation, but step (i) is different. Step (i) is changed to the following (i'), and the rest of the steps remain the same.

(i')Simulator S' receives the input $(I', F_0, f_{I'}(F_0, \cdots, F_{m-1}))$, randomly selects $F_i', i \in [1, m-1]$, such that $f_{I'}(F_0, \cdots, F_i, \cdots, F_{m-1}) = f_{I'}(F_0, \cdots, F_i', \cdots, F_{m-1}'), F_i \overset{c}{\equiv} F_i'$.

In the execution of Algorithm 2, the conspirators I' to obtain the information view: $view_0^{\pi}(F_0, f_0(F_0, \cdots, F_{m-1})) = \{F_0, Bins_{F_i}, f_{I'}(F_0, \cdots, F_{m-1})\}$, and then let $S'(F_0, f_{I'}(F_0, ..., F_{m-1})) = \{F_0, Bins_{F_i'}, f_{I'}(F_0, \ldots, F_{m-1}')\}$.

It can be seen from the above equation that I' only receives the binning information $Bins_{F_i'}$ of P_i during the execution of the Algorithm 2. P_0 has no private key sk', so the binnings cannot be decrypted, and the feature data cannot be inferred. So $Bins_{F_i'}$ and $Bins_{F_i}$ are computationally indistinguishable. Finally, $\{view_0^{\pi}(F_0, f_{I'}(F_0, \cdots, F_{m-1}))\} \overset{c}{\equiv} \{S'(F_0, f_{I'}(F_0, \cdots, F_{m-1}'))\}$. In summary, the Algorithm 1 and 2 are safe.

■

In the implementation Algorithms 1 and 2, the data provider P_i does not send any actual feature data to the data application and the data application P_0 does not send the lable data to the data provider to ensure that the data of both parties are not out of the local. Secondly, P_i only encrypts the categories of the true and false groups, compared with encrypting the feature data and sending to the data application party, Algorithms 1 and 2 not only ensure the security of data privacy, but also greatly improve the computational efficiency.

C. Maximal information coefficient

Most of the current methods focus on the application of the maximal information coefficient. In [22], maximal information coefficient is used to improve the calculation of Pearson coefficient, and in [23], k-means algorithm is improved. The objective of our method is to protect the privacy of the data in the calculation of the maximal information coefficient for federated learning, so that the data can not go out of the local.

*1) **Algorithm design**:* The specific steps of maximal information coefficient algorithm combined with homomorphic encryption scheme are Algorithm 3, which is executed by the data prividers. Algorithm 4 is a child of Algorithm 3, which is executed by the data application.

In Algorithm 3, the current region is meshed by the meshing function $line \in numpy.linspace(2, s_{max}, s+1)$, where $s, s \in [2, s_{max}+1]$ is the index of $line$ in the x direction, and the y direction is meshed by the meshing function $line \in numpy.linspace(2, r_{max}, r+1)$, where $r, r \in [2, r_{max}+1]$ is the index of the line $line$ in the y direction, and a region can be formed between two adjacent lines in the x and y directions. The $numpy.linspace()$ method of Numpy library in Python is used to divide the boundary in the direction of x axes. The funcation $numpy.linspace(2, s_{max}, s+1)$ returns data evenly spaced between 2 and s_{max} with $s+1$. $line$ is a line parallel to x and y axis.

The feature data F_i is discretized into these meshing regions. The maximal information coefficient requires that the number of meshes does not exceed $s_{max} \cdot r_{max} = n^{0.6}$, so the number of meshes does not exceed $s_{max} = n^{0.3}$ in the x direction and the number of meshes does not exceed $r_{max} = n^{0.3}$ in the y direction. From this, we can get a total of $s_{max} \cdot r_{max}$ meshing schemes, calculate the information coefficient of each meshing schemes $MIC[F_i; Y]$, and then get the maximal information coefficient.

Algorithm 3 Algorithm 3 requires the data providers P_i execution

```
Input: $P_i$ holds $F_i = \{X_0,\cdots,X_z\}, k \in [0, z-1]$,
$X_k = \{x_0,\cdots,x_{n-1}\}$, $id = \{0, 1, 2,\cdots, n-1\}$.
Output: Maximum information coefficient $MIC[F_i; Y]$.

1: for $i = 0$ to $m-1$ do
2:   for $k = 0$ to $z-1$ do
3:     $s_{max} = n^{0.3}, s \in [2, s_{max}+1]$
4:     $line \in numpy.linspace(2, s_{max}, s+1)$
5:     for $s = 2$ to $s_{max}+1$ do
6:       for $id = 0$ to $n-1$ do
7:         if $(line[s]) \leqslant x_{id} < line[s+1]$ then
8:           $region_i[id][s : s+1] = Enc(1)$
9:           $region_i[id][: s], region_i[id][s+1 :] = Enc(0)$
10:        end if
11:      end for
12:    end for
13:  end for
14:  $P_i$ sends the matrix $region_i$ to $P_0$
15:  $P_i$ obtain the $S[F_i; Y]$ by the step[17] of Algorithm 4
16:  $P_i$ decrypts the $S[F_i; Y]$ to obtain the number of each
     region, and initializes falling frequency $p$
17:  $p(F_i, Y), p(F_i), p(Y) \leftarrow Dec(S[F_i; Y]), n$
18:  $MIC[F_i; Y] \leftarrow p(F_i, Y), p(F_i), p(Y)$
19: end for
```

P_i holds $F_i = \{X_0,\cdots,X_z\}, k \in [0, z-1]$, $X_k = \{x_0,\cdots,x_{n-1}\}$, $id = \{0, 1, 2,\cdots, n-1\}$ to calculate the correlation between feature data F_i and the lable data Y. P_i uses s to traverse the meshing scheme in the x direction, where s belongs to $[2, s_{max}+1]$. The region matrix $region_i$ is used to record the fall of feature data F_i into the region. The row coordinate of the region matrix is the index of feature data and the columns of the matrix represent all regions, the number of columns in the region is $(s_{max}+1) \times (r_{max}+1)$.

Next, P_i determines whether the value of the feature data x falls between the two underscore $line[s]$ and $line[s+1]$ in the steps 7-11 of Algorithm 3, and if it falls between this interval, the id-th row, the s to $s+1$ columns of region matrix $region_i[id][s : s+1]$ is marked as $Enc(1)$. $Enc(1)$ is obtained by encrypting 1 using the (1). The other columns of region matrix $region_i$ is marked as $Enc(0)$ as the steps 9-10 of Algorithm 3. For the current region, different s and r will generate different meshing schemes, P_i computes the all meshing schemes, and sends the region matrix $region_i$ to the data application party P_0.

In Algorithm 4, P_0 obtains the region matrix $region_i$ by the step 17 of Algorithm 3. P_0 determines whether the value of the data points Y falls between the two lines $line[r]$ and $line[r+1]$ in the steps 7-9 of Algorithm 4, and if it falls between this interval, the id-th row, the $[r : r+1]$ columns of region matrix $area_i[id][r : r+1]$ is equal the $region_i[id][r : r+1]$, where $id \in [0, n-1]$. The purpose of this step 8 of Algorithm 4 is to intersect the region corresponding to (F_i, Y) data points

Algorithm 4 Algorithm 4 requires the data application P_0 execution

```
Input: $P_0$ holds $Y = \{y_0,\cdots,y_{n-1}\}, id = \{0,\cdots,n-1\}$.
Output: Falling number of data points in per region
$S[F_i; Y]$.
1: for $i = 0$ to $m-1$ do
2:   $P_0$ obtain the $region_i$ by the step[17] of Algorithm 3
3:   $r_{max} = n^{0.3}, r \in [2, r_{max}+1]$
4:   $line \in numpy.linspace(2, r_{max}, r+1)$
5:   for $r = 2$ to $r_{max}+1$ do
6:     for $id = 0$ to $n-1$ do
7:       if $(line[r]) \leqslant y_{id} < line[r+1]$ then
8:         $area_i[id][r : r+1] = region_i[id][r : r+1]$
9:       end if
10:    end for
11:  end for
12:  for $k = 0$ to $z-1$ do
13:    for $v = 0$ to $(s_{max}+1) \times (r_{max}+1)$ do
14:      $S[X_k; Y][v] = sum(area_i[: , v]), X_k \in F_i$
15:    end for
16:  end for
17:  $P_0$ sends the $S[F_i; Y]$ to $P_i$
18: end for
```

to get the exact position. Then, P_0 uses Pailliar's addition homomorphism to calculate the sum of each column of the matrix $area_i$ to get the falling number of data points in each region, the falling number of data points in each region is recorded in $S[F_i; Y]$. P_0 sends the $S[F_i; Y]$ to P_i.

*2) **Correctness analysis of Algorithms 3 and 4:*** The correctness of Algorithms 3 and 4 can be guranteed by the basic principle of Maximal information coefficient and the special coding method used in this paper.

Theorem 3. *Algorithms 3 and 4 can correctly compute the value of Maximal information coefficient.*

Proof: P_i initializes the matrix of falling frequency of data points in each region p. According to the step 8 of Algorithm 3, the region of data points is marked as $Enc(1)$, and the number of data points for each region is counted using the addition homomorphism of Paillier.

P_i decrypts $S[F_i; Y]$ to obtain the number of data points for each region $Dec(S[F_i; Y])$ by (2), such as $Dec(S[X_k; Y]) = [count_0, count_1, \cdots, count_{(s\times r)}], X_k \in F_i$. According to the number $Dec(S[F_i; Y])$ and the number of all data points n, P_i computes the falling frequency p. For example, the joint probability of region 0 is:

$$p_0(X_k, Y) = \frac{count_0}{n} = \frac{count_0}{len(X_k)} \tag{6}$$

Then calculate the joint probability $p(X_k, Y), p(X_k), p(Y)$, calculate the mutual information value $I[X_k; Y]$ through probability $p(X_k, Y), p(X_k), p(Y)$ as follows:

$$I[X_k; Y] = \sum_{X_k, Y} p(X_k, Y) log_2 \frac{p(X_k, Y)}{p(X_k)p(Y)} \tag{7}$$

and calculate $MIC[X_k;Y]$ according to the mutual information value $I[X_k;Y]$ and $p(X_k,Y), p(X_k), p(Y)$ as follows:

$$MIC[X_k;Y] = \max_{s\cdot r<(s_{max}\cdot r_{max})}\left(\frac{I(X_k;Y)}{log_2(min(s,r))}\right) \quad (8)$$

where $max()$ is the function that evaluates the maximum value, returning the maximum value in the array and min is the function that evaluates the minimum value, returning the minimum value in the array. s_{max}, r_{max} is generally taken to the power of 0.6 or 0.55 of the total data n. It can be calculated as $s_{max} \times r_{max} = n^{0.6}$. Due to paillier's additive homomorphism, according to $S[F_i;Y]$, P_i aggregates all the $MIC[X_k;Y]$ to get the Maximal information coefficient $MIC[F_i;Y]$.

■

3) ***Security analysis of Algorithms 3 and 4:*** Firstly, it is proved that the label matrix Y of P_0 is safe. In Algorithm 3, the status of all participants $P_i, i \in [1, m-1]$ is equal, and if any $m-1$ participants conspire, no information of the unconspirators region matrix can be obtained. Secondly, it is necessary to prove the security of the feature data F_i of the data provider P_i. According to the distribution region of its feature data, the participant P_i uses the $Enc(1)$ encrypted by the public key encryption of the Paillier scheme to mark the distribution region. In Algorithm 4, although P_0 received the region matrix of feature data in the calculation process, P_0 did not have the corresponding private key and cannot decrypt the region distribution of feature data F_i.

In step 14 of Algorithm 4, $S[X_k;Y]$ is calculated by summing the mark $Enc(1)$ corresponding to all data points in the region $aera_i$ using the additive homomorphism of paillier by (3). Since $Enc(1)$ is encrypted, P_0 cannot get any information of P_i from the encryption matrix $area_i$. In the step 17 of Algorithm 3, all participants P_i decrypts the number of data points in per region, and only P_i knows the decryption result. Therefore, the maximal information coefficient $MIC[F_i;y]$ of P_i is also safe.

Theorem 4. *Algorithms 3 and 4 can safely compute the maximal information coefficient in federated learning.*

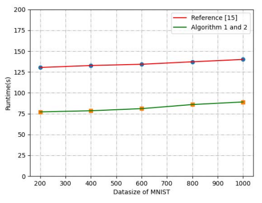
In Algorithms 3 and 4, we can get the maximal information coefficient to evaluate the feature correlation. In step 15 of Algorithm 3, the data provider P_i does not send any real feature data values to the data application but the ciphertext of region code $Enc(1)$, and the step 17 of Algorithm 4, instead of sending a label matrix Y, P_0 sends ciphertext matrix $S[F_i;Y]$, which is the number of data points in each region. From the above analysis, we have the following Theorem IV-C3, the strict proof of which is similar to the definition, omitted here.

V. Experiment

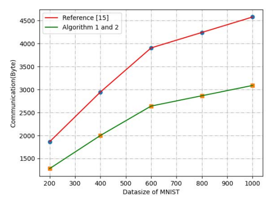
We have implementated the framework FEFL. The framework focuses on the improvement of data privacy protection and computation overhead.Experimental equipment include six PCs based on x64 processor with 16.0 GB RAM (15.4GB available), Windows10 64-bit operating system and AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz. The framework is implemented using Python language and communicate using the Flask framework. We have selected the $MNIST$ and $Spamtest$ datasets.

A. Analysis of experimental data of Algorithms 1 and 2

In terms of computing cost, compared with [14], our computation cost is greatly reduced, we only encrypt group information, and group number is far less than the feature data. The comparison results of runtime and communication are shown in Figure 2:



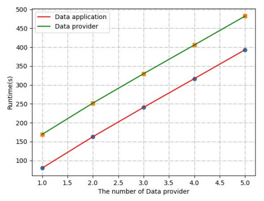
(a) runtime



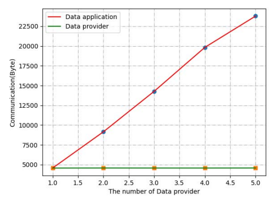
(b) communication

Figure 2. Runtime and communication of two parties in Algorithms 1 and 2

Figure 2 shows the two parties experimental results of Algorithms 1 and 2. In Figure 2(a), the abscissa is the size of $MNIST$ dataset. The vertical coordinate is the running time. In Figure 2(b), the abscissa is the size of $MNIST$ dataset, the vertical coordinate is the communication cost. It can be seen that the running time and communication cost of FEFL are significantly less than that of [14].



(a) runtime



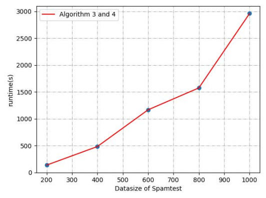
(b) communication

Figure 3. Runtime and communication of multi-party in Algorithms 1 and 2

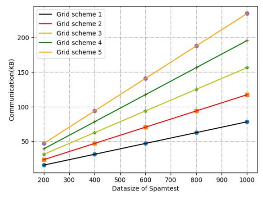
Figure 3 shows the multi-party experimental results of Algorithms 1 and 2. In Figure 3(a), the abscissa is the number of the data providers. The vertical coordinate is the running time of the data providers and data applications. In Figure 3(b), the abscissa is the number of the data providers. Because multiple data providers are computing in parallel, the running time of the data provider does not vary much.

B. Analysis of experimental data of Algorithms 3 and 4

Figure 4 shows the two parties experimental results of Algorithms 3 and 4. In Figure 4(a), the abscissa is the size of $Spamtest$ dataset. The vertical coordinate is the running time. In Figure 3(b), the abscissa is the size of $Spamtest$ dataset. The vertical coordinate is communication cost. Figure 3(b) shows different communication cost of different grid schemes in Algorithm 3.



(a) runtime



(b) communication

Figure 4. Runtime and communication of two parties in Algorithms 3 and 4

VI. Conclusion

In this paper, we have proposed and implemented a feature engineering framework called FEFL. Compared with other frameworks, it confines the feature data in locally, improving the safety and efficiency of the feature engineering in federated learning. Based on the FEFL framework, we have proposed new methods to compute Chi-square binning and the maximal information coefficient, which does not send the actual data and has stronger security.

VII. Acknowledgment

This research is supported by the National Natural Science Foundation of China (Grants: U1936120, U1636216), the National Key R&D Program of China (2017YFB0802805 and 2017YFB0801701), the Fok Ying Tung Education Foundation of China (Grant 171058), the European Union's Horizon 2020 research and innovation programme under grant agreement No. 952697 (ASSURED) and No. 101021727 (IRIS), and the Basic Research Program of State Grid Shanghai Municipal Electric Power Company (52094019007F). Daojing He is the corresponding author of this article.

References

[1] A. Kjamilji, E. Savas, and A. Levi, "Efficient secure building blocks with application to privacy preserving machine learning algorithms," *IEEE Access*, vol. 9, pp. 8324–8353, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3049216

[2] Y. Liu, J. J. Q. Yu, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7751–7763, 2020.

[3] X. Liu, H. Li, G. Xu, R. Lu, and M. He, "Adaptive privacy-preserving federated learning," *Peer-to-Peer Networking and Applications*, vol. 13, pp. 2356–2366, 2020.

[4] M. Song, Z. Wang, Z. Zhang, Y. Song, Q. Wang, J. Ren, and H. Qi, "Analyzing user-level privacy attack against federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2430–2444, 2020.

[5] A. Deac, "Regulation (eu) 2016/679 of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and the free movement of these data," *Perspectives of Law and Public Administration*, vol. 7, 2018.

[6] H. Zhu, R. S. M. Goh, and W. K. Ng, "Privacy-preserving weighted federated learning within the secret sharing framework," *IEEE Access*, vol. 8, pp. 198 275–198 284, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3034602

[7] A. C. Yao, "Protocols for secure computations (extended abstract)," in *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*. IEEE Computer Society, 1982, pp. 160–164. [Online]. Available: https://doi.org/10.1109/SFCS.1982.38

[8] X. Li, R. Dowsley, and M. D. Cock, "Privacy-preserving feature selection with secure multiparty computation," *CoRR*, vol. abs/2102.03517, 2021. [Online]. Available: https://arxiv.org/abs/2102.03517

[9] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, ser. AISec'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–11. [Online]. Available: https://doi.org/10.1145/3338501.3357370

[10] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3298981

[11] P. Fang, Z. Cai, H. Chen, and Q. Shi, "FLFE: A communication-efficient and privacy-preserving federated feature engineering framework," *CoRR*, vol. abs/2009.02557, 2020. [Online]. Available: https://arxiv.org/abs/2009.02557

[12] H. Zhang, Y. Chen, L. Xiang, H. Ma, J. Shi, and Q. Zhang, "Deep quaternion features for privacy protection," *CoRR*, vol. abs/2003.08365, 2020. [Online]. Available: https://arxiv.org/abs/2003.08365

[13] Y. Sei and A. Ohsuga, "Privacy-preserving chi-squared test of independence for small samples," *BioData Min.*, vol. 14, no. 1, p. 6, 2021. [Online]. Available: https://doi.org/10.1186/s13040-021-00238-x

[14] Y. Zhang and Z. Chen, "methods and devices for federated feature engineering data processing(in chinese)," 2020.

[15] K. Prabhu, B. Jun, P. Hu, Z. Asgar, S. Katti, and P. Warden, "Privacy-preserving inference on the edge: Mitigating a new threat model," in *Research Symposium on Tiny Machine Learning*, 2020.

[16] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, ser. Lecture Notes in Computer Science, J. Stern, Ed., vol. 1592. Springer, 1999, pp. 223–238. [Online]. Available: https://doi.org/10.1007/3-540-48910-X_16

[17] I. Moulinier, "Feature selection: A useful preprocessing step," in *19th Annual BCS-IRSG Colloquium on IR Aberdeen, UK. 8th-9th April 1997*, ser. Workshops in Computing. BCS, 1997. [Online]. Available: http://ewic.bcs.org/content/ConWebDoc/4839

[18] T. Gu, J. Guo, Z. Li, and S. Mao, "Detecting associations based on the multi-variable maximum information coefficient," *IEEE Access*, vol. 9, pp. 54 912–54 922, 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3070925

[19] J. Acharya, C. L. Canonne, and H. Tyagi, "Inference under information constraints I: lower bounds from chi-square contraction," *IEEE Trans. Inf. Theory*, vol. 66, no. 12, pp. 7835–7855, 2020. [Online]. Available: https://doi.org/10.1109/TIT.2020.3028440

[20] H. Kang, G. Liu, Z. Wu, Y. Tian, and L. Zhang, "A modified flowdroid based on chi-square test of permissions," *Entropy*, vol. 23, no. 2, p. 174, 2021. [Online]. Available: https://doi.org/10.3390/e23020174

[21] G. Sun, J. Li, J. Dai, Z. Song, and F. Lang, "Feature selection for iot based on maximal information coefficient," *Future Gener. Comput. Syst.*, vol. 89, pp. 606–616, 2018. [Online]. Available: https://doi.org/10.1016/j.future.2018.05.060

[22] Y. Tian, H. Zhang, P. Li, and Y. Li, "A complementary method of pcc for the construction of scalp resting-state eeg connectome: Maximum information coefficient," *IEEE Access*, vol. 7, pp. 27 146–27 154, 2019.

[23] R. Wang and H. Li, "Mic-kmeans: A maximum information coefficient based high-dimensional clustering algorithm," R. Silhavy, Ed. Cham: Springer International Publishing, 2019, pp. 208–218.