Fed-OLF: Federated Oversampling Learning Framework for Imbalanced Software Defect Prediction Under Privacy Protection

Xiaowen Hu, Ming Zheng, *Member, IEEE*, Rui Zhu, *Member, IEEE*, Xuan Zhang, *Member, IEEE*, and Zhi Jin, *Fellow, IEEE*

***Abstract*—Software defect prediction technology can discover potential errors or hidden defects by establishing prediction models before the use of products in the field of software engineering, so as to reduce subsequent problems and improve software quality and security. However, building predictive models requires enough software defect dataset support, especially defect samples. Due to the involvement of confidential information from various organizations or enterprises, software defect data cannot be shared and effectively utilized. Therefore, to achieve collaborative training of multiparty shared software defect prediction models while keeping the data local to various organizations, we made the federated learning framework for the issue of software defect prediction. Meanwhile, the nondefect and defect instances in software defect datasets are usually imbalanced, which can seriously affect the software defect prediction performance of the model. Therefore, this study designs a novel federated oversampling learning framework Fed-OLF. First, the TabDiT method based on deep generative model is proposed in Fed-OLF to expand and rebalance the local imbalanced software defect dataset of each client with a certain degree of privacy protection. Second, a parameter aggregation strategy based on local information entropy is proposed in Fed-OLF to further optimize the parameter aggregation effect of the global shared model, thereby achieving better model performance. We conduct extensive experiments on the PROMISE dataset and the NASA Promise repository, and experimental results on the PROMISE dataset and the NASA Promise repository show that, the proposed Fed-OLF exhibits better predictive performance under the F1-score, G-mean, and AUC metrics when compared with the advanced baseline methods. In addition, we verify that both the TabDiT method and the parameter aggregation strategy based on local information entropy in Fed-OLF are useful, and the combination of them can more effectively improve model performance.**

***Index Terms*—Federated learning (FL), imbalanced software defect dataset, oversampling, privacy protection, software defect prediction.**

Received 30 July 2024; revised 27 October 2024; accepted 26 December 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62306009, Grant 62272006, and Grant 62002310, in part by the Major Project of Natural Science Research in Colleges and Universities of Anhui Province under Grant KJ2021ZD0007, in part by Wuhu Science and Technology Bureau Project (2022jc11), in part by Science Foundation of Yunnan Jinzhi Expert Workstation under Grant 202205AF150006, in part by Yunnan Provincial Natural Science Foundation Fundamental Research Project under Grant 202101AT070004, and in part by Major Project of Yunnan Natural Science Foundation under Grant 202302AE090020030. Associate Editor: T. Takahashi. *(Corresponding author: Ming Zheng.)*

Xiaowen Hu is with the School of Computer and Information, Anhui Normal University, Wuhu 241002, China (e-mail: xwhu@ahnu.edu.cn).

Ming Zheng is with the School of Computer and Information, Anhui Normal University, Wuhu 241002, China, and also with the Anhui Provincial Key Laboratory of Industrial Intelligence Data Security, Anhui Normal Unviersity, Wuhu 240002, China (e-mail: mzheng@ahnu.edu.cn).

Rui Zhu and Xuan Zhang are with the School of Software, Yunnan University, Kunming 650091, China.

Zhi Jin is with the Key Laboratory of High Confidence Software Technologies, Ministry of Education; the School of Computer Science, Peking University, Beijing 100871, China.

Digital Object Identifier 10.1109/TR.2024.3524064

I. Introduction

SOFTWARE defect prediction technology is based on machine learning and other methods to analyze software data. By mining and learning software defect data, a software defect prediction model is constructed to predict potential defect issues in software products [1], [2]. Software defect prediction is crucial in modern industry to improve software reliability and avoid software problems during software operation [3], [4]. In recent years, various machine learning based software defect prediction models have attracted a lot of research work, and some of them have achieved encouraging results [5], [6].

A. Motivation

Despite the progress made by machine learning based models in software defect prediction, most of them still face the following challenges.

In the real world, organizations and enterprises are independent of each other, resulting in different data distributions in their software defect prediction datasets. The common data distribution differences are class distribution imbalance and quantity imbalance. Specifically, consider scenario with multiple enterprises of different scales, including large, medium-sized, and small enterprises. These enterprises are committed to developing a learning-based software defect detector that aims autonomously identify defect modules in their respective software products. However, there is a significant scale gap between these enterprises. Their datasets exhibit size imbalance, meaning the sample number in the dataset is different. In addition, the software defect datasets of enterprises also suffer from class imbalance. Although there are software defect samples in the dataset, compared to no-defect software samples, these defect samples only account for a small proportion [7], [8].

In practical applications, collecting and labeling defect data of software requires a large amount of labor and time costs, which leads to a limited number of samples of available high-quality software defect datasets [9]. It is especially difficult for small enterprises to create datasets large enough for model training.

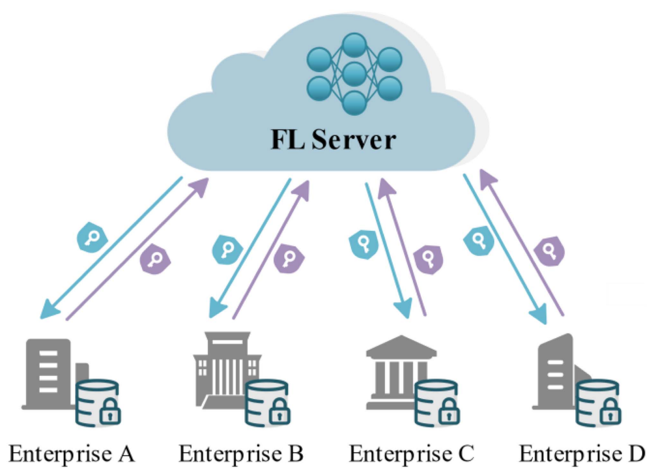

Fig. 1. Application example of FL in software defect area.

Although each organization and enterprise has a certain amount of software defect data, the amount of this data is limited. Software defect detection models based on such datasets and machine learning algorithms often have the problem of insufficient training. This will lead to insufficient prediction performance of the model and limit its practical application effect [10], [11].

In traditional centralized machine learning scenarios, organizations upload private software defect dataset to the data center, and then train model based on the data from the data center. However, software defect data is private and confidential, including code data, software version data, and other sensitive information, which involves the confidentiality of enterprises and organizations. If it is leaked or exploited in the training process, it will directly threaten the reputation and property security of organizations or enterprises. In addition, there are currently General Data Protection Regulation and California Consumer Privacy Act laws that protect personally sensitive data. Due to these strict data protection regulations and privacy policies, organizations and enterprises cannot merge data sets to train the model [12], [13]. This limitation will result in incomplete utilization of data in the field of software engineering, including software defect data.

Federated learning (FL) [14] is a distributed learning method as shown in Fig. 1. In FL framework, individual organizations or enterprises train local models on local datasets and upload model parameters to the server. The global shared model is obtained by aggregating the uploaded parameters based on the server.

Training software defect prediction models based on the FL framework can not only aggregate small-scale datasets from multiple enterprises to train the model, but also protect the data security and privacy of these enterprises. However, it cannot solve the issue of poor model prediction performance caused by imbalanced class and sizes of dataset [15], [16]. This limitation is because the server aggregates model parameters based on the size of client datasets in most FL methods. In other words, this aggregation means that the model parameters trained on a client with a large dataset have more weight and greater impact on the parameters of the global shared model.

Meanwhile, when a client has a large but severely imbalanced software defect dataset, the model training on such local dataset may cause its performance to be unsatisfactory. Because the local model parameters will have a greater impact when the server updates parameters resulting in poor performance of the global shared model [17].

This inspires us to design a novel aggregation strategy that uses an indicator to fairly measure the quality of client local software defect dataset, thereby determining the contribution of the local model parameters in aggregation. Therefore, it is beneficial and valuable to design a method to overcome the data accessibility and imbalanced software defect data distribution problems, enabling models to learn the characteristics of software defect data well while reducing the risks associated with data privacy security.

B. Contribution

To alleviate the aforementioned challenges in software defect prediction, this study proposes a new FL framework, Fed-OLF, which not only improves the performance of software defect prediction model under imbalanced data distributions, but also protects the data privacy of various organizations to a certain extent.

Specifically, akin to FedAvg [14], Fed-OLF aggregates parameters on a central server through collaborative learning between client models to obtain a globally shared model. It only shares model update parameters and data distribution information rather than sensitive local software defect data, meeting the data privacy and security requirements of enterprises. Meanwhile, a new oversampling method TabDiT based on deep generative model is proposed in Fed-OLF, which balances the client training software defect dataset by synthesizing defect samples, solving the negative impact of model performance training on imbalance software defect data.

Considering the risk of privacy leakage in the model parameters trained on the local client. In Fed-OLF, the parameters trained by the model on a balanced software defect dataset can reduce the risk of privacy breaches caused by malicious attackers analyzing the model parameters to obtain confidential information from sensitive data. Because the uploaded model parameters are trained on a balanced software defect dataset with added synthetic samples rather than the raw dataset, the attacker may not be able to infer sensitive information from the raw real software defect dataset.

In addition, Fed-OLF introduces a novel weighted aggregation strategy based on local information entropy (LEW). This strategy measures the contribution of client model parameters in aggregation based on the local information entropy, further optimizing the global shared model parameters to improve the predictive performance.

With the above-mentioned improvements, our Fed-OLF enhances the robustness of the model predictive performance on imbalanced software defect datasets, while considering the strict data privacy protection requirements of each organization. In summary, our study has the following contributions.

1) This study proposes Fed-OLF, a novel FL framework designed to improve the predictive performance of models on imbalanced software defect datasets while playing a role in data privacy protection to a certain extent.
2) A new oversampling method TabDiT has been designed in Fed-OLF to synthesize software defect samples. It not only effectively addresses the negative impact of imbalanced software defect datasets on model training during FL, but also plays a role in protecting the privacy of local model upload parameters.

3) Fed-OLF introduces a new aggregation strategy LEW based on the local information entropy, thereby enhancing the aggregation effect of the global shared model to improve the predictive performance.

The rest of this article is organized as follows. The related work is introduced in Section II. In Section III, the details of Fed-OLF are described. The experimental setup is provided in Sections IV, and V provides the experimental results. In Section VI, the limitations of our method are discussed. Section VII summarizes the research content of this study and provides prospects for future research work.

II. Related Work

A. Federated Learning

The Google AI Research Center first proposed FL [14], which is a distributed machine learning approach that supports large-scale participants. Previous studies have demonstrated the ability of FL to promote practical applications in some fields, such as autonomous driving [18], medical diagnostics [19], credit card fraud detection [20], and network intrusion detection [21], [22].

FL achieves collaborative training among multiple participants while ensuring privacy protection and data security. Specifically, FL, as a collaborative mechanism, includes a central server and multiple clients. In FL system, each client is an organization or enterprise, server randomly selects a subset $\mathbb{K}_r \subseteq K$ of clients to participate in training in each round of communication and assigns the current global shared model. The clients train models independently on their own local datasets. After each round of training, clients upload updated model parameters to a central server. The server adopts the parameter aggregation strategy to aggregate the model parameters uploaded by the client and obtain an updated global shared model. Then server sends the updated global shared model parameters to all clients, and each client optimizes the local model with the new model parameters from global shared model. FedAvg [14] is the most basic algorithm in FL. The server aggregates and updates the global shared model parameters according to the client dataset size. The FedAvg aggregation process is defined as follows:

$$W^{\mathrm{r}+1} = \sum_{k\in\mathbb{K}_r} \theta^r[k] W_k^{r+1} \tag{1}$$

where $\theta^r$is the federated aggregation vector at communication round r, which determines the contribution of the received local models and W_k^{r+1} denotes the updated model of client k. FedAvg employs the local sample size $\mid D_k \mid$ of client k as a federated aggregation vector, the weight is proportional to the local dataset size

$$\theta^r[k] = \frac{|D_k|}{\sum_{j\in\mathbb{K}_r} |D_j|}, \quad \forall k \in \mathbb{K}_r. \tag{2}$$

Since the researchers proposed FedAvg in 2017, FL has undergone significant development, allowing it to be applied to a variety of fields involving sensitive data. Recent studies applying FL to various scenarios have focused on protecting privacy and reducing communication overhead [23], [24], but only a few studies have applied FL to software defect prediction [16].

B. Imbalanced Software Defect Data Learning

In the real world, data distribution is generally imbalanced. In the problem of software defect prediction, most software only has defects in a few modules, resulting in a much larger number of nondefect samples than defect samples, forming an imbalanced software defect dataset [25]. In imbalanced software defect datasets, there are few defect samples that are insufficient to support the training of predictive models. Therefore, the oversampling method is considered for the imbalanced software defect dataset. On the one hand, the categories can be balanced to reduce the impact on model training, and on the other hand, the data scale can be expanded to provide enough training for the model.

With the rapid development of deep learning technology in recent years, deep generative models have been widely applied in different fields such as audio [26], image [27], and video synthesis [28]. The generation model of tabular data is becoming more and more important. The generated data distribution by the traditional oversampling is similar to raw data, which is easy to produce overfitting problems. The generative model has the ability of self-learning and can generate diverse samples, and its excellent generation ability has been widely used [29], [30], and some researchers have applied it to imbalanced software defect data [31]. Xu et al. [32] proposed Conditional Tabular GAN (CTGAN) and a variational autoencoder for mixed-type tabular data generation (TVAE) based on the generative adversarial networks (GAN) [33] and variational autoencoder (VAE) [34] respectively. OCT-GAN [35] is a generative model based on neural ordinary differential equations. SOS [36] and StaSy [37] are tabular data synthesis methods based on the generation mechanism of scores. TabDDPM [38] and CoDi [39] apply the popular diffusion-based generation model to generate tabular data.

Although research results continue to increase, the research on data augmentation based on generative models mainly focuses on image and video generation in the field of computer vision, and generative models applied to tabular data still have room for further improvement in synthesizing high-quality samples. In GAN, the training process is often unstable. The improved generative model based on GAN is prone to mode collapse in the training, which means that the model generates a few samples and fails to cover the diversity of data. The quality and number of training data will greatly affect the performance of the score-based generated model. When the training data is noisy or imbalanced, the generated results may also be affected. Tabular data has complex and unique features. The high dimensionality, sparsity, and class imbalance of data pose significant challenges to the denoising neural network in the generative model based on the diffusion model. The current generative model based on the diffusion model adopts simple U-Net architecture or MLP architecture as the denoising network, and such simple neural network architecture is difficult to generate high-quality tabular data.

C. Imbalanced Software Defect Data Challenge in FL

The distribution of each class in software defect datasets owned by different organizations and enterprises can vary greatly. Different sizes enterprises have different sizes datasets. In FL, the software defect data of each client is independently, which makes the local data distribution of client not consistent with the overall software defect data distribution, and there may be an obvious mismatch between local imbalance and global

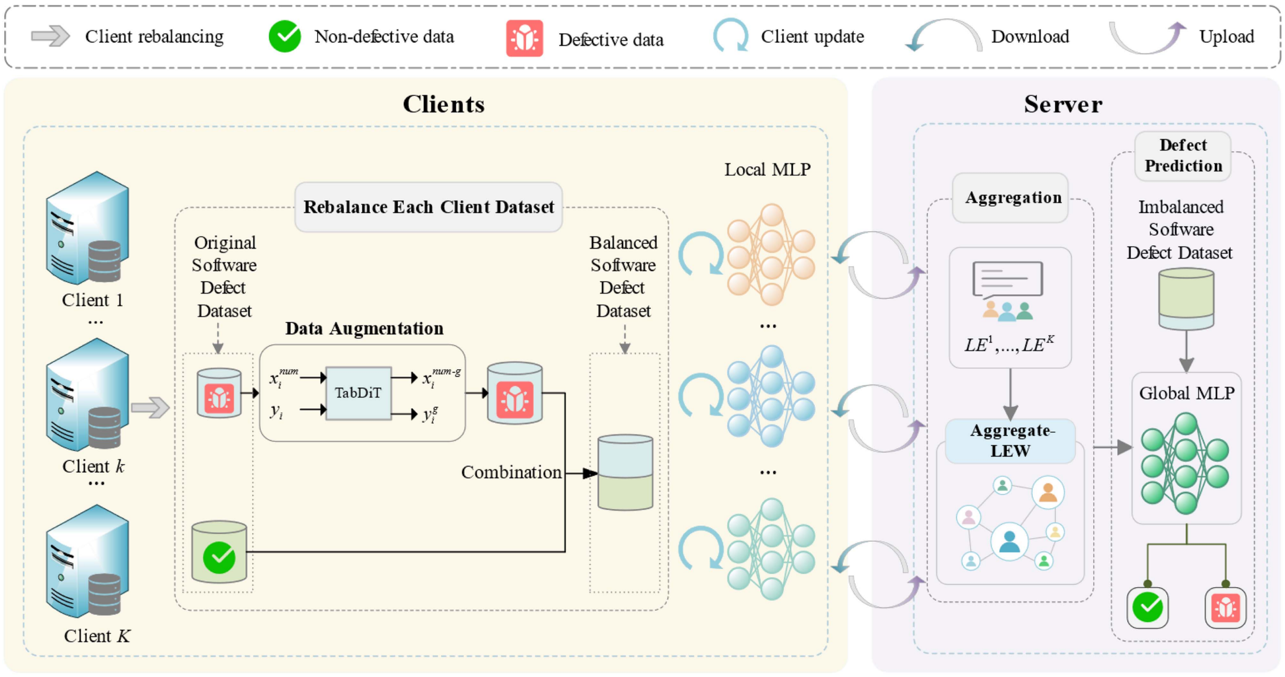

Fig. 2. Fed-OLF framework overview.

imbalance. To distinguish these imbalance problems of software defect dataset in FL, the imbalanced software defect dataset problem can be summarized into three types:

1) Local imbalance, which means that the software defect dataset distribution in each client is class imbalanced.
2) Global imbalance, from a global perspective, the collection of software defect dataset in all clients is imbalanced.
3) Size imbalance, where the software defect dataset size of each client is uneven. This study mainly focuses on local imbalance and global imbalance.

Recently, some research have focused on addressing the imbalance problem in FL. FedNoRo [40] used knowledge distillation and distance-aware aggregation function to update the federated model, and introduced logical adjustment (LA) to solve the imbalanced data. FedNoRo introduced LA to address imbalance problem, which increased the focus on the minority class of data, but also led to the neglect of majority class. Zhang et al. [41] designed an efficient heterogeneity sensing customer sampling mechanism, namely Fed-CBS, based on a class imbalance measurement index to achieve privacy protection, which can reduce the class imbalance of client packet datasets. However, in the FL practical application, the clients available in each round of communication are not mandatory, so this strategy has limited applicability by actively selecting clients. FedGR [42] proposed the imbalanced softmax function and gravitation regularizer to slove the problem of imbalanced sample number within the client and promote cooperation between clients to solve the cross-client imbalance problem. However, in the case of high imbalance and few defect samples, the ability of FedGR strategy to learn the characteristic information of defect samples is limited.

Relevant studies indicate that the final quality of deep neural network models depends on the first few training cycles. During the critical period, no matter how much additional training is conducted during this period, low-quality or insufficient training data will lead to irreversible degradation of model performance [43]. This phenomenon has been revealed in recent work in FL [44], [45], which has brought to our attention to the various imbalances in FL.

III. Proposed Fed-OLF

This study first provided an overview of Fed-OLF in this section, and then elaborated on the proposed framework and its workflow.

A. *Framework Overview*

To address the aforementioned issues in software defect prediction based on FL, we propose Fed-OLF, which oversamples the imbalanced software defect training dataset of each client to achieve class distribution balance while ensuring data security and privacy. Meanwhile, the FL aggregation process is optimized considering the different information attributes of client datasets. Fed-OLF not only reduces the interference of various imbalance problems on training and aggregation, improves the model prediction performance, but also protects the privacy of client model parameters to a certain extent.

As shown in Fig. 2, such as a typical FL process [14], the Fed-OLF framework also includes multiple clients and a central server. The client trains the local model with private software defect datasets. The central server provides collaborative learning between the global model and client devices. Prior to FL server initialization, each client uses the TabDiT method to process their own imbalanced software defect dataset in local. After each client software defect dataset is rebalanced, FL server first distributes the global shared model to each client, and the model on the client is trained using a locally balanced software defect dataset to obtain updated model parameters, which are uploaded to the server. The server optimizes the parameters according to

Algorithm 1: Training Process of Fed-OLF.

```
Input: rounds of communication: $R$, number of local
       epochs: $E$, number of clients: $K$, the fraction of
       participating clients: $c$, software defect dataset on
       client $k$: $D_k$.
Output: Global shared model $W_G^R$
1: Server executes:
2:   Initialize $W_G^0$;
3:   Collect local information entropy of clients: $LE$;
4:   for $r = 1$ to $R$ do
5:     $N_r = \max(c \cdot K, 1)$
6:     $\mathbb{K}_r \leftarrow$ (random subset of $N_r$);
7:     for $k \in \mathbb{K}_r$ parallelly do
8:       $w_k^{r+1} \leftarrow$ LocalUpdate($k$, $W_G^r$);
9:       Compute the federated aggregation vector $\theta^r[k]$
         by (15);
10:    end for
11:    Aggregate local models by (16);
12:  end for
13:  return $W_G^R$
14: Clients executes:
15:  // clients rebalancing
16:  for each client $i = 1$ to $N$ do
17:    $D_i^O \leftarrow$ TabDiT($D_i$);
18:  end for
19:    LocalUpdate($k$, $W_G^r$):
20:    $w_k^r \leftarrow W_G^r$
21:    for $e = 1$ to $E$ do
22:      $w_k^{r+1} \leftarrow w_k^r - \eta \nabla \ell(w_k^r; D_k^O)$;
23:    end for
24:    return $w_k^{r+1}$
```

the parameter aggregation strategy LEW, finally gets the updated global model, and repeats the above steps in multiple rounds of communication.

In the typical FedAvg, the imbalance problems are not solved. In Fed-OLF, a novel federated oversampling learning framework is proposed to address the imbalanced software defect dataset problem, which can not only rebalance the client training dataset but also protect the privacy of model parameters. Second, we improve the parameter aggregation strategy of the server and design a weighted aggregation strategy based on the local information entropy of the client. Combining these two components, Fed-OLF can realize the goal of sharing the global model with multiparty training while ensuring client data privacy and improving software defect predictive performance of the model.

Algorithm 1 describes the training process of Fed-OLF. After starting the FL training task, the server initializes the global shared model and initiates communication. When a new round of communication r begins, the server sends the global shared model to all participating local clients. Then, the server randomly selects a group client to participate in this training round. For each client, the local imbalanced software defect dataset is preprocessed, and the training software defect dataset is balanced by using the TabDiT method. The client participated in this training round trains the downloaded global shared model on their balanced software defect dataset and uploads updated parameters to server. Finally, the server calculates the federated aggregation vector according to the local information entropy

Algorithm 2: TabDiT.

```
Input: Client training software defect dataset: $D = \{(x_i^{\text{num}}, y_i)\}$, $i = 1, \ldots, p$; $y_i \in \{01\}$
Output: Balanced software defect dataset of client: $D^O$.
Clients:
1: Divide $D$ into nondefect and defect class: $D_{\text{nondefect}}$,
   $D_{\text{defect}}$;
2: Identify the size of $D_{\text{nondefect}}$ and $D_{\text{defect}}$: $N_{\text{nondefect}}$,
   $N_{\text{defect}}$;
3: Calculate the number of generated samples: $N_g$
   $\leftarrow N_{\text{nondefect}} - N_{\text{defect}}$;
4: TF-MLP Training ($D_{\text{defect}}$);
5: $D^{\text{gen}} \leftarrow$ TF-MLP Sampling ($N_g$);
6: $D^O \leftarrow D \cup D^{\text{gen}}$;
7: return $D^O$
```

of the client software defect dataset attribute, and then uses the parameter aggregation strategy to update global shared model. After repeated the R rounds of communication, a multiparty shared global model W_G^R is obtained.

B. Fed-OLF Workflow

The Fed-OLF workflow includes client rebalancing, initialization, model training, and parameter aggregation, as shown in Fig. 3.

1) Client Rebalancing: Before FL server initialization, the local imbalanced software defect dataset of the client participating in the FL model training task needs to be rebalanced (①) to resolve the local imbalance and global imbalance problem in FL. The local imbalance in the client data can lead to weight divergence and precision loss of model training. We propose an oversampling method TabDiT based on diffusion model [38] to solve the imbalanced software defect dataset problem in local clients, as shown in Algorithm 2.

TabDiT uses different modeling strategies to deal with various feature types in tabular data for imbalanced software defect datasets of local client. Specifically, for the software defect dataset, a tabular data sample $x = [x^{\text{num}}, y]$ contains N_{num} numerical features $x^{\text{num}} \in \mathbb{R}^{N_{\text{num}}}$ and classification feature y with C_i categories. Gaussian diffusion is employed to model the numerical attributes and multinomial diffusion is employed to model the categorical attributes. We first apply one-hot encode on the categorical features to obtain $x_{\text{ohe}}^{\text{cat}_i} \in \{0, 1\}^{C_i}$, and normalize the numerical features as the input of the model. Each individual forward diffusion process will process each categorical attribute to ensure that the noise component of each feature is sampled independently. The TabDiT algorithm first divides the imbalanced software defect dataset into nondefect and defect datasets, and then calculates the number of synthesized samples of the model (line 1–4). Considering the imbalanced software defect dataset of client, TabDiT trains on software defect datasets to build a generative model.

The forward diffusion process in TabDiT: Given defect samples $x_0 \in D_{\text{defect}}$ and the time steps T, then the forward process is x_t obtained by adding Gaussian noise to the x_{t-1} obtained by time step t, and the step size is controlled by the variance table $\{\beta_t \in (0, 1)\}_{t=1}^{T}$, then

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \varepsilon \tag{3}$$

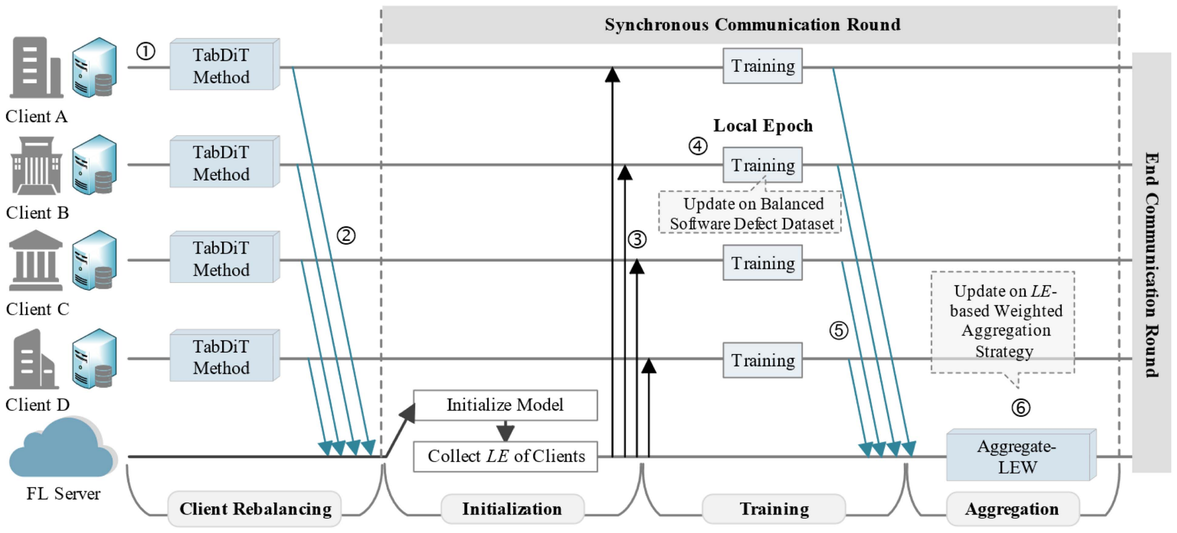

Fig. 3. Fed-OLF workflow.

where x_i denotes the sample after adding Gaussian noise in time step *i*. When $\alpha_t = 1 - \beta_t, \bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$, x_t can be expressed as

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \varepsilon \tag{4}$$

where $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$ is a Gaussian noise.

The reverse process of TabDiT: Added noise ϵ is predicted by the multilayer neural network model, and a new sample close to the original data distribution is generated by gradually removing the noise in each time step *t*

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \varepsilon_\theta(x_t, t) \right) + \sigma_t z \tag{5}$$

where α_t = 1-β_t, $\bar{\alpha}_t = \prod_{i \leq t} \alpha_i$, $z \sim \mathcal{N}(0, \mathbf{I})$, ϵ_θ (x_t, *t*) is an estimation function for predicting the real noise ϵ based on x_t [46]. The core of TabDiT method is to improve the denoising effect of model on noisy data in the reverse process, which directly affects the quality of the generated software defect samples. Therefore, in TabDiT reverse process, we use the improved Transformer architecture and combine the MLP model [47] to construct a hybrid architecture TF-MLP

$$\text{TF}-\text{MLP}(x) = \text{MLP}(\text{TF}(x)) \tag{6}$$

$$\text{TF}(x) = \text{Linear}(\text{ReLU}(\text{Transformer}(x))) \tag{7}$$

$$\text{MLP}(x) = \text{Linear}(\text{MLPBlock}(\ldots(\text{MLPBlock}(x)))) \tag{8}$$

$$\text{MLPBlock}(x) = \text{Dropout}(\text{ReLU}(\text{Linear}(x))). \tag{9}$$

The timestep *t*, class *y*, and input x_{in} are processed in the same way as [48], [49], as follows:

$$t\_emb = \text{Linear}(\text{SiLU}(\text{Linear}(\text{SinTimeEmb}(t)))) \tag{10}$$

$$y\_emb = \text{Embedding}(y) \tag{11}$$

$$x = \text{Linear}(x_{in}) + t\_emb + y\_emb \tag{12}$$

where SinTimeEmb refers to a sinusoidal time embedding as in [48] and [49] with a dimension of 128. The model trains defect samples and generates enough new samples to make the raw local imbalanced software defect dataset completely balanced (line 5–6). In the reverse process, the denoising neural network based on the TF-MLP architecture can learn the correlation between different features through the self-attention mechanism and make more accurate noise predictions. Therefore, the TabDiT method achieves a better denoising effect, resulting in the generation of high-quality software defect samples.

In addition, while realizing the balanced software defect datasets of local client, it also solves the global imbalance problem. In FL, if the condition of local client balance is satisfied, then the global balance is also satisfied, and the proof process is as follows.

We define the variables: in FL, assuming there are *K* clients, and each client *k* (*k* = 1, …, *K*) has a local imbalanced software defect dataset D_k, in which the nondefect class $C_{\text{defect}}^{k\text{non}-}$ size is represented by $N_{\text{defect}}^{k\,\text{non}-}$, and the defect class C_{defect}^{k} size is represented by N_{defect}^{k}. From a global perspective, in the set of all client local imbalanced software defect data, the sample quantity of global nondefect class $\mathbb{C}_{\text{nondefect}}$ and defect class $\mathbb{C}_{\text{defect}}$ is $\mathcal{N}_{\text{nondefect}}$ and $\mathcal{N}_{\text{defect}}$, respectively.

Proposition: In FL, if local balance is satisfied for all clients, then global balance is satisfied.

Proof: When the clients are local balanced, then

$$N_{\text{non-defect}}^{k} = N_{\text{defect}}^{k} \qquad ① \text{ for any } k\varepsilon\{1, \ldots K\}.$$

From a global perspective, for global dataset:

$$\mathcal{N}_{\text{nondefect}} = \sum_{k=1}^{K} N_{\text{nondefect}}^{k} \qquad ②$$

$$\mathcal{N}_{\text{defect}} = \sum_{k=1}^{K} N_{\text{defect}}^{k} \qquad ③$$

By substituting ① into ②, then

$$\mathcal{N}_{\text{nondefect}} = \sum_{k=1}^{K} N_{\text{nondefect}}^{k} \sum_{k=1}^{K} N_{\text{defect}}^{k} = \mathcal{N}_{\text{defect}}$$

$$\text{So } \mathcal{N}_{\text{nondefect}} = \mathcal{N}_{\text{defect}} \qquad ④.$$

As can be seen from ④, FL global balance is satisfied.

Thus, "In FL, if local balance is satisfied for all clients, then global balance is satisfied" is proved.

2) Initialization: During the initialization of the FL, the server waits for the participating client to join. The client participates in training by sending the server local information entropy value that measures the software defect data quality of the local client (②). The information entropy [50] is an metric to measure the degree of discreteness and randomness of a dataset. Suppose that the software defect dataset D contains class M and x_i belongs to class m is probability $p^m_{(x_i)}$, the information entropy is defined as

$$I\text{−Ent}(D) = -\sum_{m=1}^{M} p^m_{(x_i)} \log_2 p^m_{(x_i)}. \tag{13}$$

The smaller the information entropy value, the higher the purity of D, indicating a larger sample concentration, smaller dispersion, lower randomness, and less information content in the set. On the contrary, the larger the information entropy value, the lower the purity of set D, indicating more sample dispersion, greater uncertainty, and more information content in the set. We consider that there are two categories in software defect dataset, namely nondefect samples and defect samples. For dataset D_k of client k, the local information entropy for client k is defined as

$$\begin{aligned}\text{LE}(k) = &-\left(\frac{|D^k_{\text{nondefect}}|}{|D_k|}\right)\log_2\left(\frac{|D^k_{\text{nondefect}}|}{|D_k|}\right)\\ &-\left(\frac{|D^k_{\text{defect}}|}{|D_k|}\right)\log_2\left(\frac{|D^k_{\text{defect}}|}{|D_k|}\right)\end{aligned} \tag{14}$$

where $\mid D_k \mid$ denote the size of dataset D_k. We analyze that the larger the LE(k) value of local dataset D_k on client k, the larger the information entropy value, indicating that the greater the dispersion of the dataset, the greater the randomness, and the more the information. When all clients participating in federated training are identified, the FL server initializes the weights and optimizers of the global shared model, and then collects the LE parameters uploaded by all participating clients.

3) Model Training: When each communication round begins, the server deploys the global shared model to clients (③) and randomly selects a group of clients to participate in the FL training. Then the clients download the global shared model parameters, and participating clients perform E rounds local training on balanced local software defect dataset in each epoch with mini-batch SGD algorithm (④). In the framework, the training dataset is balanced, and the local model trained on this dataset can fully learn the information of software defect samples, without causing the model to skew, so as to focus on more important defect samples. After all participating clients complete their local training, they send updates to the FL Server (⑤).

4) Parameter Aggregation: Since each client generates software defect samples independently, the distribution of each class in software defect dataset of different the local clients will vary greatly. So, we propose a new parameter-weighted aggregation strategy, which takes into account the dispersion degree and information difference of each client raw imbalanced software defect dataset, to measure the model parameters contribution degree of each local client in the aggregation. The local parameters collected by FL server are aggregated according to the local information entropy uploaded by the client (⑥), and the weight is proportional to the value of local information entropy.

TABLE I
EXPERIMENTAL NOTATIONS

Notation	Description
R	Communication round
η	Learning rate
B_t	Test batch size
E	Local epochs
B	Local batch size
K	Total number of clients
c	The fraction of clients participating in each training round
α_d	Dirichlet distribution coefficient

In communication r, the federated aggregation vector based on LE is defined as

$$\theta^r[k] = \frac{\text{LE}(k)}{\sum_{j\in\mathbb{K}_r}\text{LE}(j)} \quad \forall k \in \mathbb{K}_r. \tag{15}$$

Finally, FL server aggregates all local model parameters involved in training and updates the global shared model to

$$W_G^{r+1} = \sum_{k\in\mathbb{K}_r} \theta^r[k] w_k^{r+1}. \tag{16}$$

The global model parameter update process consists of two steps. First, the server calculates federated aggregation vector based on the local information entropy of the participating client to determine the weight of the local parameter aggregation. Next, the parameters W_G^{r+1} of global shared model are calculated according to (16) for next training round.

IV. Experimental Setup

This section introduces the experimental setup, including imbalanced software defect datasets, compared baselines, environmental setup, evaluation metrics, statistical tests and privacy. The symbols used in the experiment are listed in Table I.

A. Datasets

We selected eight imbalanced software defect datasets from different Java projects in PROMISE dataset [16] and eight defect prediction datasets from the NASA Promise repository for experiments. In addition, all methods adopted ten-fold cross-validation on each software defect dataset in the experiments, taking the average of ten results as the comprehensive evaluation of model prediction performance to reduce random errors. The imbalanced software defect dataset information is shown in Table II.

B. Baseline and Comparison Methods

We combined the TabDiT method proposed in Fed-OLF and various existing tabular data generation methods into the FedAvg framework for experimental comparison.

1) CTGAN [32] is the most popular and well-known GAN-based synthetic data generation model.
2) OCT-GAN [35] is a neural network consisting of ordinary differential equations.
3) TabDDPM [38] is a simple design of DDPM for tabular problems.
4) CoDi [39] solves the training challenge caused by mixed data types by adopting a double-diffusion model approach.

TABLE II
INFORMATION OF PROMISE AND NASA DATASETS

Dataset	Feature	Class	Sample size	Defect rate (%)	IR	Dataset	Feature	Class	Sample size	Defect rate (%)	IR
ant-1.7	20	Non-defect	579	22.28	3.49	CM1	21	Non-defect	449	9.84	9.16
		Defect	166					Defect	49		
camel-1.4	20	Non-defect	727	16.63	5.01	JM1	21	Non-defect	6110	21.49	3.65
		Defect	145					Defect	1672		
camel-1.6	20	Non-defect	777	19.48	4.13	KC1	21	Non-defect	1783	15.46	5.47
		Defect	188					Defect	326		
ivy-1.2	20	Non-defect	312	11.36	7.80	KC2	21	Non-defect	415	20.50	3.88
		Defect	40					Defect	107		
jedit-4.2	20	Non-defect	319	13.08	6.65	MC1	38	Non-defect	1942	2.31	42.22
		Defect	48					Defect	46		
poi-2.0	20	Non-defect	277	11.78	7.49	PC1	21	Non-defect	1032	6.94	13.40
		Defect	37					Defect	77		
xalan-2.4	20	Non-defect	613	15.21	5.57	PC3	37	Non-defect	943	12.44	7.04
		Defect	110					Defect	134		
xerces-1.3	20	Non-defect	384	15.23	5.57	PC4	37	Non-defect	1280	12.21	7.19
		Defect	69					Defect	178		

TABLE III
PARAMETER SETTINGS ON DIFFERENT DATASETS

Dataset	Client Parameter			Dataset	Client Parameter		
	N	α_d	c		N	α_d	c
ant-1.7	5	1.0	1.0	CM1	5	2.0	1.0
camel-1.4	5	2.0	1.0	JM1	15	1.5	1/3
camel-1.6	5	2.0	1.0	KC1	5	2.0	1.0
ivy-1.2	5	2.0	1.0	KC2	5	2.0	1.0
jedit-4.2	5	2.0	1.0	MC1	5	2.0	1.0
poi-2.0	5	2.0	1.0	PC1	5	1.0	1.0
xalan-2.4	5	2.0	1.0	PC3	5	2.0	1.0
xerces-1.3	5	2.0	1.0	PC4	5	2.0	1.0

5) STaSy [37] is a score-based tabular data synthesis generation model.

In addition, we also compare the typical parameter aggregation strategy in FedAvg with the LEW strategy and record the typical parameter aggregation strategy in FedAvg as AVG.

C. Environmental Setup

In Fed-OLF, the cross entropy is used as a loss function and communication rounds R is 300, learning rate η of SGD is 0.1 as the optimizer for all optimization processes, and test batch size B_t to 512. For local training, E and B is 5 and 16, respectively. For heterogeneous data distribution among clients, Dirichlet distribution is used to divide the data. The MLP is used as global shared model and local model, each model has two fully connected layers, where the ReLU activation function connects the input layer and the hidden layer. All comparison methods applied the same model structure. The FL framework and MLP are implemented in PyTorch. The parameters related to the client for each imbalanced software defect dataset are shown in Table III.

D. Performance Evaluation Metrics

To accurately evaluate the performance of the method, we use the F1-score, G-mean and the area under curve (AUC) based on the receiver operating characteristic (ROC) curve, which are widely used in imbalanced data study to measure the model classification performance [51], [52].

E. Statistical Tests

To further discuss and evaluate the effectiveness of Fed-OLF objectively, statistical tests are used to determine whether there are significant differences across all methods on different software defect datasets. The statistical tests process consists of three steps. First, on each software defect dataset, all methods are ranked from best to worst according to the test performance of each method, and the order values 1, 2, …, the best method order value is 1; if the test performance of any of the methods is the same, the order value is equally divided. The Friedman test is then used to determine whether these methods all perform identically. If not, then the hypothesis that all methods perform identically is rejected. Finally, Nemenyi posthoc test needs to be used for checking whether there is a significant difference between any two methods. In this study, the confidence level $\alpha = 0.05$.

F. Privacy

To study Fed-OLF in a privacy-related context, we measure the generated software defect samples privacy with mean distance to closest record (DCR) [38]. Specifically, we calculate the minimum L2 distance between each synthesized software defect sample and the real data. Average DCR takes the average of these distances for all the generated defect samples. A low DCR value indicates that the sample synthesized by the generated model mimics some real data points in nature, which may result in a violation of privacy requirements. A higher DCR value indicates that the generation model can generate a "new" software defect data, rather than just an approximate repetition of the real software defect data that already exists.

V. EXPERIMENTAL RESULTS

To verify the effectiveness of the proposed Fed-OLF, we analyzed the experimental results by addressing four research questions.

1) *RQ-1*: Does the Fed-OLF method outperform baseline on various software defect datasets?
2) *RQ-2*: In Fed-OLF, is the data privacy generated by TabDiT better than the baseline?
3) *RQ-3*: Does each component in Fed-OLF improve model predictive performance?

A. *RQ-1: Does the Fed-OLF Method Outperform Baseline on Various Software Defect Datasets?*

Results: Tables IV and V present all the experimental results. The following conclusions can be analyzed and drawn by observing the experimental results.

1) In FL framework with the same AVG aggregation strategy, our TabDiT approach can improve the software defect predictive performance compared to other baselines.
Table IV presents the experimental results on the PROMISE dataset. When using the FL framework with the same AVG aggregation strategy, our TabDiT method outperforms other baselines on all eight PROMISE datasets. For example, on the camel-1.6 dataset, compared with other baselines, the TabDiT increases 8.42%–12.25%, 7.4%–11.32%, and 2.78%–5.38% in the F1-score, G-mean, and AUC, respectively. On the jedit-4.2 dataset, TabDiT increases by 6.11%, 6.41%, and 4.31% in F1-score, G-mean, and AUC, respectively, compared with TabDDPM that performs best in the baseline methods.
Table V presents the experimental results on the NASA dataset. Our TabDiT outperforms other baselines on CM1, JM1, KC1, PC1, PC3, and PC4 datasets when using an FL framework with the same AVG aggregation strategy. For example, on the CM1 dataset, the TabDiT method increases 5.91%–25.39%, 11.23%–47.73%, and 4.11%–12.16% over other baselines in metrics, respectively. On the PC3 dataset, our TabDiT increases 3.77%, 2.78%, and 2.01% in metrics, compared with TabDDPM that performs best in the baseline methods.
Therefore, according to the above-mentioned analysis, TabDiT method is superior to other baselines in improving model predictive performance in FL frameworks with AVG aggregation strategy.
2) In FL framework using the same LEW aggregation strategy, our TabDiT method can improve the software defect predictive performance compared to other baselines.
Table IV presents the experimental results on the PROMISE dataset. When using the same LEW aggregation strategy in the FL framework, our TabDiT method outperforms other baseline methods on all 8 PROMISE datasets. For example, on ant-1.7 dataset, TabDiT increases 1.73%–5.24%, 0.76%–7.02%, and 0.81%–3.36% in F1-score, G-mean, and AUC, respectively, compared to other baselines. On xalan-2.4 dataset, TabDiT outperforms CoDi, with improvements of 4.41%, 4.19%, and 2.52% in F1-score, G-mean, and AUC, respectively.
Table V presents the experimental results on the NASA dataset. When using the same LEW aggregation strategy in the FL framework, our TabDiT method outperforms other baseline methods on the CM1, JM1, KC1, KC2, MC1, PC1, and PC3 datasets. For example, on CM1 dataset, TabDiT increases by 6.13%–30.53% on G-mean and 0.79%–7.65% on AUC compared to other baselines. On the JM1, KC1, KC2, MC1, and PC3 datasets, our TabDiT method outperforms other baseline methods to varying degrees in F1-score, G-mean, and AUC metrics, showing the best performance.
Therefore, based on the above-mentioned analysis, in the FL framework using LEW aggregation strategy, the TabDiT method outperforms other baselines in improving model predictive performance.
3) When using the same baseline method, our proposed LEW aggregation strategy can achieve better parameter aggregation performance compared to AVG aggregation strategy, thereby improving the model prediction performance.
Tables IV and V indicate that on all datasets, using the LEW strategy can improve model performance to varying degrees compared to the AVG strategy when using the same baseline method. For example, in Table IV, using the same OCT-GAN method on xerces-1.3 dataset, our LEW strategy can improve performance by 11.22%, 11.29%, and 3.76% in F1-score, G-mean, and AUC, respectively, compared to the AVG strategy. In Table IV, using the same TabDiT method on xerces-1.3 dataset, our LEW strategy can improve performance by 4.92%, 4.17%, and 3% in F1-score, G-mean, and AUC, respectively, compared to AVG strategy. In Table V, using the same CoDi method on CM1 dataset, our LEW strategy can improve F1-score, G-mean, and AUC by 10.09%, 10.87%, and 5.33%, respectively, compared to AVG strategy. In Table V, using the same TabDiT method on JM1 dataset, our LEW strategy can improve F1-score, G-mean, and AUC by 9.62%, 10.27%, and 2.93%, respectively, compared to the AVG strategy.
The above-mentioned results indicate that when using the same baseline method, combining our LEW aggregation strategy within the FL framework can effectively improve the software defect prediction performance of the model.
4) In general, our Fed-OLF method, which combines TabDiT and LEW aggregation strategies, can effectively handle local and global imbalances of FL and train the best defect predictive model.
As shown in Tables IV and V, the combination of TabDiT and LEW aggregation strategies performs better in most datasets than the combination of any other baseline method and aggregation strategy. Thorough analysis showed that the TabDiT method can generate higher quality samples compared to other baseline methods. Each client can train the model on a balanced and high-quality software defect dataset. In addition, our LEW aggregation strategy can further optimize parameter updates on the server. In the same number of training rounds, LEW strategy can aggregate model parameters according to the information entropy characteristics of each client dataset, which can achieve higher model performance than the traditional data number-based aggregation strategy.
5) *Statistical Test Results.*
We used statistical tests to further validate the significance of the differences in detection performance between our Fed-OLF method and other methods.

We combined six baselines and two aggregation strategies respectively to obtain 12 methods, among which the combination of TabDiT and LEW is our Fed-OLF method. The values of Friedman test on the F1-score, G-mean and AUC are 24.200, 33.854, and 28.748, respectively. The critical value of F test when confidence $\alpha = 0.05$ is 1.847. On the F1-score, G-mean and AUC, the values of Friedman test are all greater than the critical value of F test. Therefore, the hypothesis that the performance of all methods is the same is rejected. To express the performance significance difference of each algorithm more intuitively, Nemenyi post-hoc test is used to further distinguish each method. Fig. 4 shows the results visually, where (a), (b), and (c) represent the results on the F1-score, G-mean, and AUC,

TABLE IV
RESULTS OF MODEL PERFORMANCE ON PROMISE DATASET

Dataset	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
ant-1.7	AVG	F1-score (%)	44.20±7.75	46.63±9.67	49.24±8.19	47.15±7.43	43.70±8.70	49.11±5.46
		G-mean (%)	58.65±7.24	60.85±8.10	60.95±6.60	58.95±5.63	55.77±7.18	63.44±3.96
		AUC (%)	64.22±4.04	65.78±5.66	66.93±4.20	65.70±3.80	64.06±4.34	67.03±3.15
	LEW (Ours)	F1-score (%)	49.61±5.44	51.73±5.46	50.35±9.17	49.25±8.30	48.22±7.62	**53.46±7.62**
		G-mean (%)	66.27±6.61	66.33±4.94	64.45±6.88	61.77±8.06	60.07±6.7	**67.09±6.04**
		AUC (%)	68.61±4.34	69.03±3.56	67.81±5.30	67.35±4.14	66.48±4.06	**69.84±4.88**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
camel-1.4	AVG	F1-score (%)	16.72±15.43	18.85±11.49	19.66±14.09	18.72±12.23	15.49±10.73	28.77±9.98
		G-mean (%)	28.61±22.24	33.67±14.87	32.42±19.26	30.88±17.71	28.93±14.38	44.56±9.67
		AUC (%)	53.32±6.76	54.34±4.87	54.89±5.74	54.60±5.32	53.89±4.40	58.08±4.52
	LEW (Ours)	F1-score (%)	21.15±12.86	19.11±11.44	21.39±11.59	25.42±13.21	24.31±14.06	**32.54±7.56**
		G-mean (%)	37.65±13.90	34.47±16.21	35.95±16.12	39.79±12.46	37.77±16.51	**48.59±6.15**
		AUC (%)	54.99±5.79	54.75±4.94	55.52±5.43	57.00±5.87	56.88±6.03	**59.60±3.83**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
camel-1.6	AVG	F1-score (%)	16.01±10.38	19.50±12.09	19.84±7.62	18.58±8.18	19.55±8.08	28.26±11.41
		G-mean (%)	31.15±15.56	31.26±16.99	35.07±7.99	33.31±7.46	34.06±7.78	42.47±10.42
		AUC (%)	52.24±2.79	54.84±3.85	54.30±2.75	53.78±3.47	54.43±3.41	57.62±4.70
	LEW (Ours)	F1-score (%)	19.51±13.52	24.5±12.71	22.93±8.38	26.55±9.77	24.90±10.64	**30.69±11.39**
		G-mean (%)	36.75±16.46	37.79±15.5	38.12±9.21	40.76±8.44	39.61±10.14	**45.24±10.11**
		AUC (%)	53.01±5.71	55.86±5.50	55.49±2.88	57.08±3.98	56.54±4.17	**58.57±5.06**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
ivy-1.2	AVG	F1-score (%)	25.84±20.49	26.89±19.20	28.33±21.41	30.98±18.72	26.98±24.2	27.29±20.77
		G-mean (%)	43.29±32.19	39.79±27.21	43.97±30.46	43.27±23.07	37.19±31.33	45.86±32.08
		AUC (%)	61.10±16.24	59.45±9.29	61.30±12.02	60.75±7.79	60.10±10.98	62.66±12.84
	LEW (Ours)	F1-score (%)	28.4±18.77	29.63±21.85	30.26±22.43	**32.93±19.35**	31.35±27.6	31.82±22.60
		G-mean (%)	46.99±26.24	40.32±27.64	45.54±32.31	45.27±24.27	40.82±34.29	**48.84±33.23**
		AUC (%)	61.23±11.64	59.94±10.3	61.56±16.09	61.34±9.60	62.93±12.91	**65.04±13.33**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
jedit-4.2	AVG	F1-score (%)	28.03±13.21	41.92±21.92	40.65±15.37	42.36±17.38	31.79±19.24	46.76±14.25
		G-mean (%)	47.32±19.26	55.09±22.26	59.62±14.68	56.85±14.08	42.10±22.85	66.03±14.07
		AUC (%)	59.71±7.66	66.30±10.4	66.47±10.04	65.66±9.38	60.18±8.45	70.78±9.55
	LEW (Ours)	F1-score (%)	36.29±10.79	48.06±19.28	44.59±13.74	42.74±15.88	38.18±12.94	**48.36±12.32**
		G-mean (%)	56.24±13.46	64.75±15.18	65.68±13.47	57.05±13.95	53.70±12.52	**71.19±13.32**
		AUC (%)	64.46±8.36	70.24±10.55	70.15±9.07	65.96±9.10	63.80±7.92	**74.02±10.01**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
poi-2.0	AVG	F1-score (%)	13.00±15.95	17.50±23.7	27.44±21.68	32.32±22.74	21.08±17.43	33.23±20.86
		G-mean (%)	22.39±27.77	23.51±29.58	41.33±29.67	44.18±24.62	32.86±27.17	49.73±27.22
		AUC (%)	53.70±7.19	54.04±10.96	58.26±14.33	60.62±10.57	55.73±8.47	62.71±12.06
	LEW (Ours)	F1-score (%)	22.51±14.35	24.83±22.86	28.68±14.04	33.27±20.39	26.87±18.98	**37.40±21.53**
		G-mean (%)	44.07±24.38	35.70±31.43	49.73±19.52	44.39±24.48	38.91±27.01	**49.91±26.81**
		AUC (%)	57.81±9.45	58.66±13.20	59.74±9.93	61.16±10.19	59.38±9.50	**63.82±11.47**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
xalan-2.4	AVG	F1-score (%)	24.49±16.18	31.34±15.19	32.18±8.92	28.97±17.42	24.12±14.71	35.28±15.91
		G-mean (%)	37.72±21.18	45.03±17.33	50.69±7.88	40.32±21.68	39.00±16.58	50.60±13.35
		AUC (%)	56.99±6.92	58.92±7.97	59.89±4.73	58.62±7.15	56.61±6.04	61.03±8.01
	LEW (Ours)	F1-score (%)	25.53±12.82	33.83±16.95	33.55±11.03	36.30±14.92	34.89±18.45	**40.71±18.06**
		G-mean (%)	43.92±13.66	51.51±21.09	50.24±9.60	52.48±11.95	48.51±19.80	**56.67±15.43**
		AUC (%)	57.12±6.83	61.76±9.28	60.34±5.69	61.79±7.54	61.52±8.15	**64.31±10.41**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
xerces-1.3	AVG	F1-score (%)	40.82±12.75	39.41±21.44	43.83±19.88	40.24±16.41	49.42±16.00	47.74±7.17
		G-mean (%)	58.27±13.37	53.59±28.08	59.19±22.25	56.53±14.91	62.02±12.67	65.34±6.11
		AUC (%)	65.33±8.64	65.93±11.07	67.48±10.37	64.91±8.88	68.74±8.19	69.10±4.26
	LEW (Ours)	F1-score (%)	42.09±12.56	50.63±17.09	45.05±14.31	45.02±15.48	51.56±18.27	**52.66±12.88**
		G-mean (%)	59.79±12.70	64.88±11.71	63.48±12.18	60.75±13.14	65.86±14.98	**69.51±7.69**
		AUC (%)	66.26±7.31	69.69±9.03	67.99±8.15	67.18±8.62	71.27±10.24	**72.10±6.63**

TABLE V
RESULTS OF MODEL PERFORMANCE ON NASA DATASET

Dataset	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
CM1	AVG	F1-score (%)	12.95±20.46	0.00±0.00	19.48±14.69	17.67±13.74	8.72±13.99	25.39±15.8
		G-mean (%)	19.82±25.49	0.00±0.00	36.50±25.58	33.66±22.88	13.69±20.98	47.73±26.31
		AUC (%)	53.38±7.96	49.11±1.09	57.16±7.72	55.47±5.91	51.92±5.10	61.27±10.52
	LEW (Ours)	F1-score (%)	16.48±11.23	14.20±20.96	26.63±22.04	**27.76±20.39**	18.29±20.87	27.21±15.46
		G-mean (%)	32.33±22.04	20.13±25.76	42.18±30.02	44.53±25.67	26.36±27.54	**50.66±21.85**
		AUC (%)	54.94±5.58	53.94±8.14	60.60±12.54	60.80±11.07	55.05±9.69	**61.59±10.72**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
JM1	AVG	F1-score (%)	26.95±11.2	19.50±9.12	15.74±5.52	12.84±3.51	11.56±3.66	19.16±4.04
		G-mean (%)	32.74±17.10	33.12±8.18	30.58±6.93	26.47±3.66	24.97±4.26	33.87±4.42
		AUC (%)	51.52±1.33	54.28±2.27	52.99±1.47	52.96±1.08	52.54±1.10	54.30±1.12
	LEW (Ours)	F1-score (%)	26.52±7.07	28.58±4.33	21.37±7.50	23.49±6.81	21.05±6.35	**28.78±4.04**
		G-mean (%)	42.60±7.69	43.97±4.54	36.29±8.30	38.25±7.16	35.95±6.71	**44.14±4.76**
		AUC (%)	55.89±2.39	57.14±1.69	54.97±2.32	55.69±2.08	54.87±2.14	**57.23±1.51**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
KC1	AVG	F1-score (%)	29.96±8.68	25.21±7.46	26.07±8.58	28.90±10.01	29.36±9.38	31.66±5.14
		G-mean (%)	47.02±11.84	39.82±7.30	42.53±9.18	43.33±9.12	44.44±9.06	47.40±5.23
		AUC (%)	59.69±4.15	57.15±2.98	57.27±3.89	58.50±4.02	58.80±3.89	59.60±2.35
	LEW (Ours)	F1-score (%)	33.30±6.30	30.85±8.26	30.92±8.62	33.92±8.2	32.98±7.28	**34.74±5.66**
		G-mean (%)	48.82±5.92	45.05±7.30	47.57±8.67	49.85±8.25	48.76±6.96	**50.78±5.23**
		AUC (%)	60.31±2.74	59.15±3.52	59.44±4.36	60.83±4.02	60.22±3.16	**61.03±2.62**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
KC2	AVG	F1-score (%)	47.80±10.70	47.41±19.87	53.38±13.22	49.72±11.24	48.16±11.02	53.29±11.52
		G-mean (%)	61.22±9.95	61.12±18.9	67.67±11.67	61.90±9.06	60.99±8.87	66.92±10.13
		AUC (%)	67.18±5.59	68.54±12.73	71.34±8.31	67.72±5.93	66.90±5.46	70.73±7.03
	LEW (Ours)	F1-score (%)	49.68±10.55	48.94±9.11	54.93±12.02	56.32±7.0	49.71±16.69	**56.38±12.8**
		G-mean (%)	64.37±9.68	64.39±9.14	69.15±9.82	69.91±7.02	62.17±13.22	**70.91±12.51**
		AUC (%)	68.55±7.25	68.36±6.25	71.97±7.84	72.63±5.35	68.06±8.97	**73.90±9.06**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
MC1	AVG	F1-score (%)	2.82±5.96	1.82±5.45	13.29±15.77	8.21±12.58	6.86±13.95	10.44±18.15
		G-mean (%)	12.77±19.56	4.41±13.24	33.99±36.72	13.37±20.42	9.46±18.96	21.85±33.95
		AUC (%)	49.70±3.94	50.18±2.97	60.65±16.64	52.64±4.6	51.76±4.75	56.36±13.91
	LEW (Ours)	F1-score (%)	4.68±5.17	12.36±13.05	13.59±11.89	7.25±11.59	5.58±8.84	**13.86±11.76**
		G-mean (%)	28.07±29.41	22.73±22.78	44.93±33.66	15.78±24.95	15.43±23.91	**45.83±33.30**
		AUC (%)	55.46±8.81	54.48±5.40	63.03±16.61	53.29±7.69	53.19±6.13	**63.40±15.23**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
PC1	AVG	F1-score (%)	9.80±12.09	18.07±14.04	25.68±12.46	24.05±12.13	24.42±16.62	25.32±9.00
		G-mean (%)	20.78±21.30	29.23±20.23	52.75±13.39	37.80±15.01	35.43±19.74	53.61±14.57
		AUC (%)	50.62±6.14	55.56±5.15	61.64±6.83	57.71±4.71	57.72±5.96	62.74±7.82
	LEW (Ours)	F1-score (%)	19.17±8.75	24.55±15.35	25.98±13.59	**30.4±15.43**	24.71±9.81	24.51±8.09
		G-mean (%)	45.11±18.38	48.04±19.35	56.88±10.70	49.47±13.86	45.78±9.93	**58.14±12.94**
		AUC (%)	58.29±9.00	60.15±8.18	62.65±6.92	61.81±7.65	59.03±4.92	**63.77±8.26**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
PC3	AVG	F1-score (%)	16.18±12.98	18.43±7.94	29.26±5.76	14.45±8.41	20.37±12.31	33.03±9.95
		G-mean (%)	31.39±22.60	37.47±9.15	50.92±6.86	26.99±14.28	36.65±16.34	53.70±8.81
		AUC (%)	53.91±5.52	54.10±3.95	59.75±3.38	53.59±2.48	55.60±5.80	61.76±5.51
	LEW (Ours)	F1-score (%)	19.99±14.00	23.38±9.88	31.22±6.96	20.12±7.67	23.24±15.84	**33.56±9.20**
		G-mean (%)	37.01±21.14	42.97±11.28	53.38±8.09	36.80±7.93	39.48±17.75	**55.61±8.53**
		AUC (%)	56.91±8.21	56.77±5.25	61.16±4.16	55.22±3.21	56.45±7.06	**62.29±5.27**
	Aggregation	Method	CTGAN	OCT-GAN	TabDDPM	CoDi	STaSy	TabDiT (Ours)
PC4	AVG	F1-score (%)	31.23±13.07	32.00±8.65	37.93±7.22	39.50±7.21	42.18±6.01	44.08±11.55
		G-mean (%)	47.35±12.72	48.01±8.18	54.39±6.36	54.16±6.71	55.13±4.63	56.70±8.68
		AUC (%)	60.54±7.33	60.40±3.75	63.39±3.58	63.77±3.58	64.47±2.52	65.53±5.10
	LEW (Ours)	F1-score (%)	43.48±8.18	44.48±9.20	43.81±10.49	42.94±8.31	**48.59±8.61**	46.59±9.11
		G-mean (%)	59.14±8.72	60.75±6.85	59.96±9.03	55.46±6.95	**63.17±6.55**	61.38±6.51
		AUC (%)	66.51±4.84	67.04±4.12	66.85±5.94	64.89±3.73	**68.87±4.20**	67.66±4.30

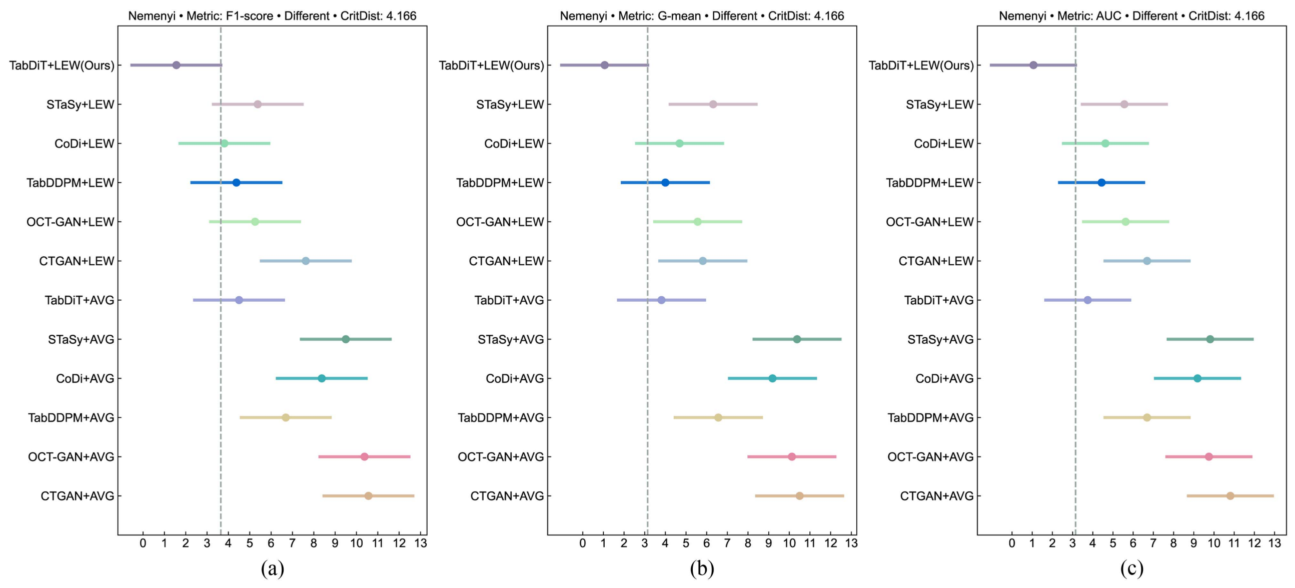

Fig. 4. Results of Nemenyi posthoc test in terms of (a) F1-score, (b) G-mean, and (c) AUC.

TABLE VI
Results of Significant Differences Between Fed-OLF and Other Methods

Metric	Fed-OLF performs better than the method	Fed-OLF significantly better than the method
F1-score	all other	CTGAN+AVG, OCT-GAN+AVG, TabDDPM+AVG, CoDi+AVG, STaSy+AVG, CTGAN+LEW
G-mean	all other	CTGAN+AVG, OCT-GAN+AVG, TabDDPM+AVG, CoDi+AVG, STaSy+AVG, CTGAN+LEW, OCT-GAN+LEW, STaSy+LEW
AUC	all other	CTGAN+AVG, OCT-GAN+AVG, TabDDPM+AVG, CoDi+AVG, STaSy+AVG, CTGAN+LEW, OCT-GAN+LEW, STaSy+LEW

TABLE VII
Comparison in Terms of Mean DCR

Oversampling Method	PROMISE Dataset								NASA Dataset							
	ant-1.7	camel-1.4	camel-1.6	ivy-1.2	jedit-4.2	poi-2.0	xalan-2.4	xerces-1.3	CM1	JM1	KC1	KC2	MC1	PC1	PC3	PC4
CTGAN	0.862	0.753	0.708	0.967	0.907	0.920	0.885	0.929	0.489	0.099	0.384	0.302	0.985	0.308	0.713	0.748
OCT-GAN	1.304	1.187	1.148	1.471	1.356	1.322	1.365	1.378	1.159	0.550	1.112	0.730	1.736	0.957	1.364	1.383
TabDDPM	0.792	0.901	0.852	0.647	0.671	0.604	1.129	0.977	0.255	0.345	0.572	0.488	0.863	0.273	0.915	1.288
CoDi	1.361	1.320	1.357	1.461	1.292	1.34	1.439	1.408	1.180	0.764	1.262	0.791	2.115	1.151	1.717	1.832
STaSy	1.187	1.323	1.328	1.087	1.08	0.956	1.465	1.400	0.881	0.680	1.243	0.879	1.335	0.957	1.157	1.634
TabDiT(Ours)	1.221	1.370	1.362	1.110	1.106	0.998	1.517	1.441	0.882	0.688	1.257	0.884	1.390	0.967	1.165	1.671

respectively. In Table VI, we summarize the methods that Fed-OLF outperforms and significantly outperforms. Thus, these results show that Fed-OLF method exhibits excellent performance in different software defect datasets.

To sum up, we conduct comprehensive experiments on various software defect datasets to evaluate the effectiveness of proposed Fed-OLF in training software prediction models. The experimental results demonstrate that Fed-OLF can improve the software defect predictive model performance on almost all software defect datasets, which proves the effectiveness of our method. The reason why Fed-OLF performs better is that TabDiT first solves the local imbalanced software defect dataset problem of clients, allowing the model of clients to train on balanced software defect datasets, especially in highly imbalanced environments, while also addressing the global imbalanced software defect dataset problem at the same time. In addition, the LEW weighted aggregation strategy can further optimize parameter aggregation process, thereby improving the prediction performance of the global shared model.

B. RQ-2: In Fed-OLF, is the Data Privacy Generated by TabDiT Better Than the Baseline?

Table VII shows the average DCR values for TabDiT, CTGAN, OCT-GAN, TabDDPM, CoDi, and StaSy. The average DCR value is used to measure the privacy of the new samples synthesized by the generative model, and the higher value is better. However, it should be noted that out-of-distribution data such as random noise will also provide a high average DCR, so it is necessary to combine the average DCR with model predictive performance on F1-score, G-mean, and AUC metrics. In Table VII, we observed that TabDiT is more private than CTGAN, TabDDPM, and StaSy, but less private than OCT-GAN and CoDi. But from the model performance in Tables IV and V, under the same aggregation strategy, the baselines based on OCT-GAN and CoDi performed poorly.

In summary, based on all the above-mentioned results, the Fed-OLF proposed in this study, on the one hand, the new samples synthesized by TabDiT have a certain degree of privacy.

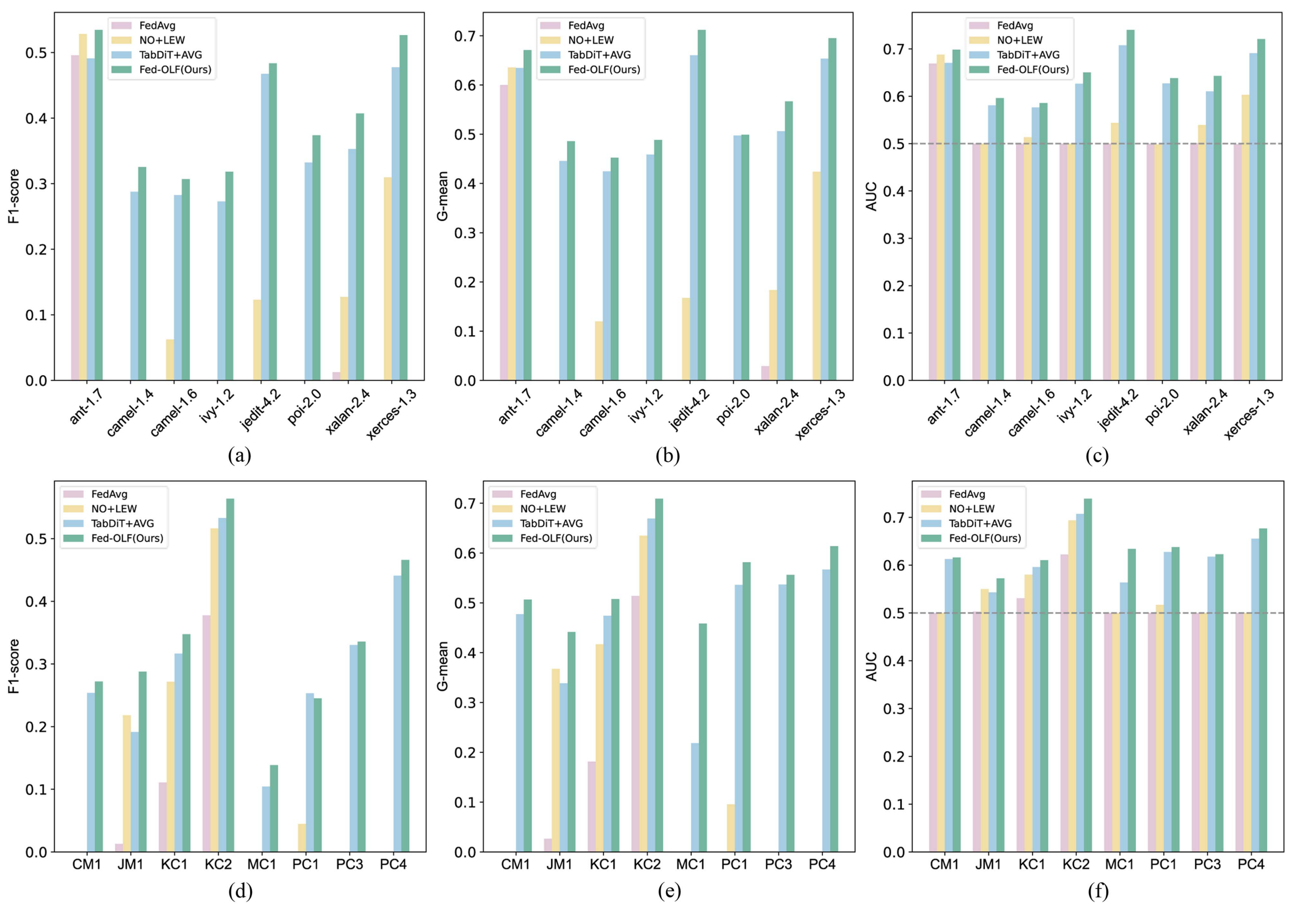

Fig. 5. Results of Ablation experiment in terms of (a) F1-score, (b) G-mean, (c) AUC of PROMISE dataset, and (d) F1-score, (e) G-mean, (f) AUC of NASA Promise repository.

On the other hand, training client model on balanced dataset can achieve better performance and better balance between data availability and privacy.

C. RQ-3: Does Each Component in Fed-OLF Improve Model Predictive Performance?

Fig. 5 shows the experimental results. The effectiveness of each component in Fed-OLF is verified by the ablation experiments. The FedAvg method and its aggregation strategy AVG are compared experimentally. The combination of NO and LEW indicates that only the LEW parameter aggregation strategy is used in the FL framework. The combination of TabDiT and AVG indicates that TabDiT is added to the FL framework and the AVG strategy is adopted.

By observation of experimental results on all datasets, we can find that on camel-1.4, camel-1.6, ivy-1.2, jedit-4.2, poi-2.0, xerces-1.3, CM1, MC1, PC1, PC3, and PC4 datasets, the values of F1-score and G-mean of FedAvg method are 0, and the values of AUC are 0.5, when the local imbalance problem of the client and global imbalance problem is not processed. It indicates that the model cannot identify defect samples, and the trained classification model tends to favor nondefect samples, and all samples are predicted to be nondefect samples. In this situation, the model tends to make random prediction. We analyze that on 16 different datasets, our Fed-OLF achieved higher model performance than methods with only one component, TabDiT or LEW. Therefore, the ablation results of all datasets show that each component of Fed-OLF is effective in improving the prediction performance of the model.

To sum up, based on the analysis of above-mentioned results, each component in Fed-OLF can improve model performance and each component is meaningful. TabDiT proposed in Fed-OLF solves the local and global imbalanced software defect data problem and expands the software defect dataset scale at the same time, which makes the local model can train on the balanced software defect dataset. In addition, a new parameter weighted aggregation strategy LEW is proposed to further optimize the effect of parameter aggregation and improve the model performance in FL. Therefore, it is necessary and beneficial to combine the two strategies.

VI. Discussion

In this section, we have a full discussion of Fed-OLF, exploring the problems that our approach can solve in cross-enterprise software defect prediction scenarios. In addition, we discuss the limitations of Fed-OLF.

For cross-enterprise software defect scenarios, Fed-OLF can be well applied in such areas where data is deficient and sensitive. Fed-OLF proposed the TabDiT method on the basis of FL framework to solve the local imbalance and global imbalance problems of software defect dataset in enterprises. In addition, the LEW-based aggregation strategy proposed in Fed-OLF can alleviate the influence of size imbalance on aggregation. It can also play a certain privacy protection role for enterprise datasets. On the one hand, the uploaded model parameters are trained on the balanced software defect dataset rather than the raw software defect dataset, and the attacker may not be able to infer the sensitive information from the raw software defect dataset. Fed-OLF only needs to upload the local information entropy value instead of the number of samples in the client dataset Therefore, Fed-OLF not only involves a small amount of dataset information transmission, which has a certain protection effect on model parameters, but also can improve the model predictive performance on software defect dataset.

There are still some limitations in Fed-OLF. It fails to eliminate the size imbalance problem in FL. In software defect prediction scenario of FL, further research is needed to investigate the negative impact of differences dataset size among enterprise on the aggregation stage. In addition, TabDiT can achieve a balance between data availability and privacy, but it cannot determine whether the generated data satisfies the privacy-aware applications in real. Therefore, the privacy issue of data generated by TabDiT needs to be further studied.

VII. Conclusion and Future Work

To make full use of sensitive data in the software defect prediction and to overcome imbalanced software defect data problem, this study proposes a novel federated oversample learning framework named Fed-OLF. First, the TabDiT method proposed in Fed-OLF solves the local imbalance and global imbalance problems in the datasets of various enterprises and organizations. Under the premise of satisfying data privacy and security, TabDiT solves the imbalanced software defect data distribution of the client by synthesizing software defect samples. The balanced software defect dataset enables the model to be fully trained while protecting the privacy of the model parameters. Second, the weighted aggregation strategy LEW in Fed-OLF effectively alleviates the negative impact of the size imbalance on the aggregation effect. This strategy further optimizes the aggregation effect of client upload parameters in aggregation stage, thus improving the software defect prediction performance of the model. In addition, extensive experimental results show that better model prediction performance can be achieved in the Fed-OLF framework compared to the baseline approach. At the same time, the ablation experiments verify that each component in Fed-OLF is effective. It is important to note that we provide an FL framework that is not limited by any prediction model, that is, the framework can support any more advanced software defect prediction model, and the client-weighted aggregation strategy is also universal.

In the future, we will further study the size imbalance problem in FL. The application of FL to other key areas of data privacy is worth investigating. It is also important to study the model parameter attack and privacy protection based on FL transmission to cloud server, which can better protect the private data of each client.

References

[1] Z. M. Zain, S. Sakri, and N. H. A. Ismail, “Application of deep learning in software defect prediction: Systematic literature review and meta-analysis,” *Inf. Softw. Technol.*, vol. 158, Jun. 2023, Art. no. 107175.

[2] G. Giray, K. E. Bennin, Ö. Köksal, Ö. Babur, and B. Tekinerdogan, “On the use of deep learning in software defect prediction,” *J. Syst. Softw.*, vol. 195, Jan. 2023, Art. no. 111537.

[3] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, “A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools,” *Eng. Appl. Artif. Intell.*, vol. 111, May 2022, Art. no. 104773.

[4] E. Iannone, R. Guadagni, F. Ferrucci, A. De Lucia, and F. Palomba, “The secret life of software vulnerabilities: A large-scale empirical study,” *IEEE Trans. Softw. Eng.*, vol. 49, no. 1, pp. 44–63, Jan. 2023.

[5] L. Song and L. L. Minku, “A procedure to continuously evaluate predictive performance of just-in-time software defect prediction models during software development,” *IEEE Trans. Softw. Eng.*, vol. 49, no. 2, pp. 646–666, Feb. 2023.

[6] M. Aniche, E. Maziero, R. Durelli, and V. H. Durelli, “The effectiveness of supervised machine learning algorithms in predicting software refactoring,” *IEEE Trans. Softw. Eng.*, vol. 48, no. 4, pp. 1432–1450, Apr. 2022.

[7] P. Manchala and M. Bisi, “Diversity based imbalance learning approach for software fault prediction using machine learning models,” *Appl. Soft Comput.*, vol. 124, Jul. 2022, Art. no. 109069.

[8] Y. Gao, Y. Zhu, and Y. Zhao, “Dealing with imbalanced data for interpretable defect prediction,” *Inf. Softw. Technol.*, vol. 151, Nov. 2022, Art. no. 107016.

[9] Y. Ren, B. Liu, and S. Wang, “Joint instance and feature adaptation for heterogeneous defect prediction,” *IEEE Trans. Rel.*, vol. 73, no. 1, pp. 741–756, Mar. 2024.

[10] S. K. Pandey, R. B. Mishra, and A. K. Tripathi, “Machine learning based methods for software fault prediction: A survey,” *Expert Syst. Appl.*, vol. 172, Jun. 2021, Art. no. 114595.

[11] Y. Yang, X. Xia, D. Lo, and J. Grundy, “A survey on deep learning for software engineering,” *Assoc. Comput. Machinery Comput. Surv.*, vol. 54, no. 10s, pp. 1–73, 2022.

[12] L. Nurgalieva, A. Frik, and G. Doherty, “A narrative review of factors affecting the implementation of privacy and security practices in software development,” *Assoc. Comput. Machinery Comput. Surv.*, vol. 55, no. 14s, pp. 1–27, Dec. 2023.

[13] O. Amaral, S. Abualhaija, D. Torre, M. Sabetzadeh, and L. C. Briand, “AI-enabled automation for completeness checking of privacy policies,” *IEEE Trans. Softw. Eng.*, vol. 48, no. 11, pp. 4647–4674, Nov. 2022.

[14] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[15] R. Malhotra and K. Lata, “Handling class imbalance problem in software maintainability prediction: An empirical investigation,” *Front. Comput. Sci.*, vol. 16, no. 4, 2022, Art. no. 164205.

[16] Y. Yang et al., “Federated learning for software engineering: A case study of code clone detection and defect prediction,” *IEEE Trans. Softw. Eng.*, vol. 50, no. 2, pp. 296–321, Feb. 2024.

[17] G. Wang, C. X. Dang, and Z. Zhou, “Measure contribution of participants in federated learning,” in *Proc. IEEE Int. Conf. Big Data*, 2019, pp. 2597–2604.

[18] S. K. Lo, Q. Lu, C. Wang, H.-Y. Paik, and L. Zhu, “A systematic literature review on federated machine learning: From a software engineering perspective,” *Amer. Chem. Soc. Comput. Surv.*, vol. 54, no. 5, pp. 1–39, 2021.

[19] Z. Chen, C. Yang, M. Zhu, Z. Peng, and Y. Yuan, “Personalized retrogress-resilient federated learning toward imbalanced medical data,” *IEEE Trans. Med. Imag.*, vol. 41, no. 12, pp. 3663–3674, Dec. 2022.

[20] W. Zheng, L. Yan, C. Gou, and F.-Y. Wang, “Federated meta-learning for fraudulent credit card detection,” in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, pp. 4654–4660.

[21] X. Huang, J. Liu, Y. Lai, B. Mao, and H. Lyu, “EEFED: Personalized federated learning of execution&evaluation dual network for CPS intrusion detection,” *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 41–56, 2023.

[22] P. Tian, Z. Chen, W. Yu, and W. Liao, “Towards asynchronous federated learning based threat detection: A DC-Adam approach,” *Comput. Secur.*, vol. 108, Sep. 2021, Art. no. 102344.

[23] J. Gao et al., “Secure aggregation is insecure: Category inference attack on federated learning,” *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 1, pp. 147–160, Jan./Feb. 2023.

[24] J. Zhou et al., “A differentially private federated learning model against poisoning attacks in edge computing,” *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 3, pp. 1941–1958, May/Jun. 2023.

[25] J. Chen, J. Xu, S. Cai, X. Wang, H. Chen, and Z. Li, “Software defect prediction approach based on a diversity ensemble combined with neural network,” *IEEE Trans. Rel.*, vol. 73, no. 3, pp. 1487–1501, Sep. 2024.

[26] H. Liu et al., “AudioLDM: Text-to-audio generation with latent diffusion models,” in *Proc. 40th Int. Conf. Mach. Learn.*, 2023, pp. 21450–21474.

[27] C. Saharia et al., "Photorealistic text-to-image diffusion models with deep language understanding," in *Proc. Neural Inf. Process. Syst.*, 2022, pp. 36479–36494.

[28] A. Blattmann et al., "Align your latents: High-resolution video synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 22563–22575.

[29] S. Zheng and N. Charoenphakdee, "Diffusion models for missing value imputation in tabular data," 2022, *arXiv:2210.17128*.

[30] M. Hernandez, G. Epelde, A. Alberdi, R. Cilla, and D. Rankin, "Synthetic data generation for tabular health records: A systematic review," *Neurocomputing*, vol. 493, pp. 28–45, 2022.

[31] S. S. Rathore, S. S. Chouhan, D. K. Jain, and A. G. Vachhani, "Generative oversampling methods for handling imbalanced data in software fault prediction," *IEEE Trans. Rel.*, vol. 71, no. 2, pp. 747–762, Jun. 2022.

[32] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Proc. Neural Inf. Process. Syst.*, 2019, pp. 7333–7343.

[33] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[34] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.

[35] J. Kim, J. Jeon, J. Lee, J. Hyeong, and N. Park, "OCT-GAN: Neural ODE-based conditional tabular GANs," in *Proc. Web Conf.*, 2021, pp. 1506–1515.

[36] J. Kim et al., "SoS: Score-based oversampling for tabular data," in *Proc. 28th Amer. Chem. Soc. SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 762–772.

[37] J. Kim, C. Lee, and N. Park, "Stasy: Score-based tabular data synthesis," in *Proc. 11th Int. Conf. Learn. Representations*, 2023, pp. 1–27.

[38] A. Kotelnikov, D. Baranchuk, I. Rubachev, and A. Babenko, "TABDDPM: Modelling tabular data with diffusion models," in *Proc. 40th Int. Conf. Mach. Learn.*, 2023, pp. 17564–17579.

[39] C. Lee, J. Kim, and N. Park, "Codi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis," in *Proc. 40th Int. Conf. Mach. Learn.*, 2023, pp. 18940–18956.

[40] N. Wu, L. Yu, X. Jiang, K.-T. Cheng, and Z. Yan, "FedNoRo: Towards noise-robust federated learning by addressing class imbalance and label noise heterogeneity," in *Proc. 32nd Int. Joint Conf. Artif. Intell.*, 2023, pp. 4424–4432.

[41] J. Zhang et al., "Fed-CBS: A heterogeneity-aware client sampling mechanism for federated learning via class-imbalance reduction," in *Proc. 40th Int. Conf. Mach. Learn.*, 2023, pp. 41354–41381.

[42] S. Guo et al., "FedGR: Federated learning with gravitation regulation for double imbalance distribution," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2023, pp. 703–718.

[43] S. Jastrzebski et al., "Catastrophic fisher explosion: Early phase fisher matrix impacts generalization," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 4772–4784.

[44] G. Yan, H. Wang, and J. Li, "Seizing critical learning periods in federated learning," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 8788–8796.

[45] G. Yan, H. Wang, X. Yuan, and J. Li, "DeFL: Defending against model poisoning attacks in federated learning via critical learning periods awareness," in *Proc. Assoc. Advance. Artif. Intell. Conf. Artif. Intell.*, 2023, pp. 10711–10719.

[46] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Neural Inf. Process. Syst.*, 2020, pp. 6840–6851.

[47] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," in *Proc. Neural Inf. Process. Syst.*, 2021, pp. 18932–18943.

[48] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," in *Proc. Neural Inf. Process. Syst.*, 2021, pp. 8780–8794.

[49] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 8162–8171.

[50] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[51] Y. Yan, Y. Zhu, R. Liu, Y. Zhang, Y. Zhang, and L. Zhang, "Spatial distribution-based imbalanced undersampling," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 6376–6391, Jun. 2023.

[52] T. Zhu, X. Liu, and E. Zhu, "Oversampling with reliably expanding minority class regions for imbalanced data learning," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 6167–6181, Jun. 2023.

Xiaowen Hu received the B.E. degree in computer science and technology, in 2022, from Anhui Normal University, Wuhu, China, where she is currently working toward the M.E. degree in computer science and technology with the School of Computer and Information.

Her current research interests include software engineering, data mining, and federated learning.

Ming Zheng (Member, IEEE) received the Ph.D. degree in information and communication engineering from Yunnan University, Kunming, China, in 2020.

He is currently an Associate Professor with the School of Computer and Information, Anhui Normal University. His research fields include imbalanced data mining and software defect prediction.

Rui Zhu (Member, IEEE) received the Ph.D. degree in software engineering from Yunnan University, Kunming, China, in 2016.

He is currently the Head and an Associate Professor of Artificial Intelligence with the School of Software, Yunnan University. His current research interests include software engineering, intelligent transportation systems, blockchain, and deep learning.

Xuan Zhang (Member, IEEE) received the Ph.D. degree in system analysis and integration from Yunnan University, Kunming, China, in 2014.

She is currently a professor of the School of Software at Yunnan University. She is the author of four books and more than 120 articles. She has been principal investigator for 16 national, provincial, and private grants and contracts. She is the core scientist of Yunnan Key Laboratory of Software Engineering and Yunnan Software Engineering Academic Team. Her current research interests include software engineering, blockchain, knowledge graphs, and natural language processing.

Zhi Jin (Fellow, IEEE) received the B.S. degree in computer science from Zhejiang University, Hangzhou, China, in 1984, the M.S. and Ph.D. degrees in computer science from the Changsha Institute of Technology, Changsha, China, in 1987 and 1992, respectively.

She is currently a Professor of computer science with Peking University, where she is the Deputy Director of the Key Laboratory of High Confidence Software Technologies (Ministry of Education). Her research interests include software engineering, requirements engineering, knowledge engineering, and machine learning.

Dr. Jin was the recipient of the IEEE TCSVC Distinguished Leadership Award, the ACM Distinguished Paper Awards (four times), and published three monographs. She serves as an Associate Editor for IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, IEEE TRANSACTIONS ON RELIABILITY, ACM TRANSACTIONS ON AUTONOMOUS AND ADAPTIVE SYSTEMS, *Empirical Software Engineering*, and *Requirements Engineering*. She is also a Fellow of CCF and AAIA.