Towards Privacy Preserving Cross Project Defect Prediction with Federated Learning

Hiroki Yamamoto †, Dong Wang †*, Gopi Krishnan Rajbahadur ‡, Masanari Kondo †,
Yasutaka Kamei †, Naoyasu Ubayashi †

Kyushu University, Fukuoka, Japan†
Huawei Technologies Canada Co., Ltd., Canada‡
Email: h.yamamoto@posl.ait.kyushu-u.ac.jp, d.wang@ait.kyushu-u.ac.jp, gopi.krishnan.rajbahadur1@huawei.com,
kondo@ait.kyushu-u.ac.jp, kamei@ait.kyushu-u.ac.jp, ubayashi@ait.kyushu-u.ac.jp

***Abstract*—Defect prediction models can predict defects in software projects, and many researchers study defect prediction models to assist debugging efforts in software development. In recent years, there has been growing interest in Cross Project Defect Prediction (CPDP), which predicts defects in a project using a defect prediction model learned from other projects' data when there is insufficient data to construct a defect prediction model. Since CPDP uses other projects' data, data privacy preservation is one of the most significant issues. However, prior CPDP studies still require data sharing among projects to train models, and do not fully consider protecting project confidentiality. To address this, we propose a CPDP model *FLR* employing *federated learning*, a distributed machine learning approach that does not require data sharing. We evaluate *FLR*, using 25 projects, to investigate its effectiveness and feature interpretation. Our key results show that *first*, *FLR* outperforms the existing privacy-preserving methods (i.e., LACE2). Meanwhile, the performance is relatively comparable to the conventional methods (e.g., supervised and unsupervised learning). *Second*, the results of the interpretation analysis show that scale-related features have a common effect on the prediction performance of the *FLR*. In addition, further insights demonstrate that parameters of federated learning (e.g., learning rates and the number of clients) also play a role in the performance. This study is served as a first step to confirm the feasibility of the employment of federated learning in CPDP to ensure privacy preservation and lays the groundwork for future research on applying other machine learning models to federated learning.**

***Index Terms*—Defect Prediction, Cross Project, Privacy Preservation, Federated Learning**

I. Introduction

Defect prediction models, a technique to support debugging work by predicting the probability of defects in a software project, are broadly regarded as a cornerstone for quality assurance [1, 2]. In recent years, cross-project defect prediction (CPDP) has gained considerable attention. CPDP is a solution to relieve the pain of collecting sufficient data in practice, where the training data is not only from the target projects but also from other projects [3].

However, CPDP is not the silver bullet and suffers from several criticized problems. One of the fundamental challenges is the lack of availability of quality transferable data [4]. For instance, knowledge of one project cannot be directly and completely transferred to another project due to a significant distribution discrepancy between them [5]. Another significant problem is selecting the most appropriate and sufficient training data. Most of the research done in CPDP make use of public and large open-source data (e.g., PROMISE [6]), since proprietary projects usually own limited historical data, especially for a new project. While, due to the differences between the development environment of open-source projects and proprietary projects, it becomes risky to use historical data of open-source projects to create quality predictors for proprietary projects from a practical point-of-view [7].

To enable the successful adoption of CPDP to proprietary projects, it is important to have models trained on similar data. However, two main barriers exist. Privacy concern of data owner is a major barrier for proprietary projects to share data, i.e., preventing the disclosure of specific sensitive metric values of the original project data [8]. Another barrier is concerning the need for the utility of the privatized source project data in CPDP, such as homogenous feature sets [9]. Proprietary projects differ from user requirements and test software modules in practice, hence, there usually exist different metrics in the data. Several researches have been carried out to address these barriers in proprietary projects. To name a few, Peters and Menzies [10] introduced MORPH, a privacy algorithm designed to reduce the attacker's trust in the released data. Peters et al. [11] proposed LACE2 which reduces the amount of data shared by using multi-party data sharing and their results showed that LACE2 yields higher privacy than the prior approach without damaging predictive efficacy.

Although these attempts, we notice that the limitation of privacy concerns cannot be fully relieved since sharing data among multiple participants is inevitable in the prior work. Recently, Federated Learning (FL) was proposed, a distributed machine learning approach that enables training on a large corpus of decentralized data residing on devices [12]. One of its well-recognized benefits is that FL enables proprietary projects to share data in a "closed-loop system" to build a common and powerful model [13]. Inspired by its successful applications in other domains (e.g., artificial intelligence [14, 15]) and its confidential feature, we conjecture that FL would be an appropriate candidate of CPDP for the aim of privacy preservation. Hence, in this study, we propose a federated learning based CPDP by incorporating a Logistic Regression model, namely *Federated Logistic Regression (FLR)*. To evaluate its performance, we conduct an empirical study on 25 open-source and close-source projects. Two research questions are formulated in terms of the effectiveness and the interpretation of *FLR*:

- **(RQ1) How well do federated learning CPDP models perform compared to traditional privacy preserved CPDP methods?**
  Motivation: Our proposed *FLR* tackles the criticized barrier of privacy concerns in CPDP, but its performance remains unknown. Thus, as an initial attempt, we would like to compare the performance of *FLR* against the (i) privacy-preserving methods (i.e., LACE2) and (ii) conventional CPDP methods (i.e., supervised and unsupervised learning).
  Results: The evaluation results on the one hand show that our proposed *FLR* model outperforms the existing privacy-preserving methods (i.e., LACE2), yielding higher precision and AUC scores. On the other hand, compared to the conventional baseline methods, the *FLR* model does not perform as well as the conventional methods, but still shows comparable prediction potential. In addition, the statistical tests confirm that the *FLR* model is likely to be assigned to the higher rank group in prediction performance.
- **(RQ2) What features affect the CPDP model using federated learning?**
  Motivation: Interpretation of defect prediction models reveals what features have a significant impact on defect rates, which is used to gain insight into avoiding defects and improve software quality more effectively. Therefore, in this RQ, we would like to determine which features impact the predictive performance of federated learning based CPDP models (*FLR*).
  Results: The results of the feature interpretation analysis demonstrate that although the most important feature differs from the evaluated datasets, we observe that scale-related features commonly have an effect on the prediction performance of the *FLR* model in terms of the top-3 features.

Our contributions are three-fold: (I) this study takes a first step towards exploring the feasibility of employing federated learning to CPDP models, in order to tackle the challenge of data privacy preservation; (II) through the evaluation results, the proposed *FLR* model outperforms the existing supervised learning that applies LACE2 in terms of higher precision and is comparable to the conventional models with low false alarm; (III) our in-depth relationship and feature impact analysis reveal insight into the characteristics of the models using federated learning, which could guide the future research direction in the context of federated learning usage in CPDP.

The remainder of this paper is organized as follows: Section II presents the background and related work in terms of cross-project defect prediction, privacy preservation, and federated learning. Section III describes the experimental design including the studied datasets, the proposed model, and approaches to address research questions. Section IV describes the results of proposed RQs. Section V further discusses the performance of our proposed model. Section VI discloses the threats to the validity of our study. Finally, we conclude the paper and provide future directions in Section VII. Our replication package is publicly available [16].

II. Background and Related Work

In this section, we introduce the background and the related literature to motivate our study.

A. Cross Project Defect Prediction (CPDP)

To relieve the limitation of insufficient defect information (especially for software companies), cross-project defect prediction (CPDP) that builds a prediction model using data from other projects has been widely studied. There are two threads of work: one is using supervised methods where the training data requires the labels, and another one is using unsupervised methods where the training data do not need to be labeled [17]. *For the supervised methods*, in the early time, Canfora et al. [18] proposed a multi-objective approach based on a multi-objective logistic regression model using a genetic algorithm. Panichella et al. [19] introduced a combined approach, coined as CODEP, and found that the superior prediction accuracy was achieved by CODEP when compared to stand-alone defect predictors. Later, Zhang et al. [20] investigated seven composite algorithms and reported that several algorithms outperform CODEP which combines different and complementary classifiers learned by different machine learning algorithms. Gong et al. [21] studied the class-imbalance problem and proposed a class-imbalance learning approach. *For the unsupervised methods*, Nam and Kim [22] proposed two novel approaches, CLA and CLAMI, showing the potential for defect prediction on unlabeled datasets in an automated manner. Specifically, the CLAMI approach evaluated on seven open-source projects led to promising prediction performances. Zhang et al. [23] examined the two types of unsupervised classifiers (i.e., distance-based classifiers and connectivity-based classifiers) for cross-project prediction. They found that a connectivity-based unsupervised classifier can compete with supervised classifiers. Zhou et al. [8] studied defect prediction using unsupervised learning Manualdown, and their results showed that Manualdown defect prediction model had equal or better prediction performance than several existing CPDP models.

Although the field of CPDP shows promise, its major component (data sharing) raises the concern of privacy preservation dramatically.

B. Privacy Preservation in CPDP

The data used for defect prediction in projects is said to be sensitive due to privacy preservation issues, and most companies are reluctant to share data that is confidential information [24]. For instance, Weyuker et al. [25] doubted that she will ever make the AT&T data public to be used

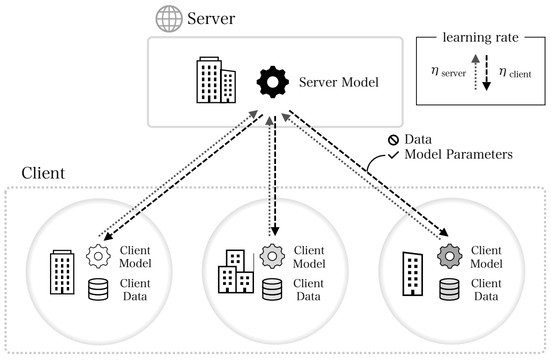


Fig. 1. The overview of federated learning

for defect predictions. Hence, privacy-preserving data-sharing becomes an important focus in CPDP. Several researches have attempted to address the privacy-preserving data-sharing issue. As one of the pioneering works, Peters and Menzies [10] proposed MORPH, a privacy algorithm designed to reduce the attacker's trust in the released data. Their results show the possibility to secure against attackers while preserving the relationships required for effective defect prediction. In the following study, Peters et al. [26] explored privatization algorithms and introduced CLIFF, an instance pruner that deletes irrelevant examples. They tested CLIFF+MORPH among 10 defect datasets in a cross-company defect prediction, and observed that CLIFFed+MORPHed algorithms provide more privacy than the state-of-the-art privacy algorithms and perform significantly better in terms of utility. Later, to mitigate the limitation of CLIFFed+MORPHed approach where it only considers a single-party scenario, Peters et al. [11] further proposed LACE2, a multi-party privacy policy. The evaluation results showed that LACE2 is comparatively less expensive than the single-party approach and LACE2 yields higher privacy than the prior approach without damaging predictive efficacy. In the recent work, Fan et al. [27] conducted a thorough empirical investigation into the utility of existing privacy-preserving data-sharing algorithms and reported ManualDown has a comparable or even better defect prediction performance. Moreover, they suggested that researchers should use ManualDown as a baseline model for comparison to develop practical privacy-preserving data-sharing algorithms.

With the use of these privacy algorithms, researchers have found that it is the potential to reduce the confidential threat during data sharing in CPDP. However, the privacy concern yet cannot be fully resolved since sharing data among multiple participants is inevitable in the existing work.

C. Federated Learning

Federated learning (FL) is a machine setting first introduced in 2016 [28], where a group of Clients collaboratively train a model under the orchestration of a central Server. Figure 1 illustrates a scenario of federated learning, in which a model is provided in a centralized Server and several Clients distributively use their local data to train the Server model. Client's individual data itself cannot be identified, because the training results of all the Client's data are shared as weights (a.k.a., *differential privacy*). Then, model weights are exchanged multiple times between the Server and Client models. Such a privacy protection mechanism effectively prevents private information from being leaked during data sharing [29, 30].

FL has recently received significant interest and is successfully adopted in research and applied perspectives [31]. For instance, FL opens up new research directions for mobile and artificial intelligence. To name a few, Hard et al. [32] applied the federated algorithm to detect mobile keyboard and confirmed that the feasibility and benefit of training models on Clients. Hao et al. [15] proposed an efficient and privacy-enhanced federated learning (PEFL) scheme for industrial AI and demonstrated the superiority of PEFL in terms of accuracy and efficiency. Liu et al. [33] developed an online visual object detection platform powered by FL, which achieves significant efficiency improvement.

Considering the privacy protection mechanism of FL and its successful application, we conjecture that FL would be an appropriate candidate to be injected into CPDP which will greatly relieve the privacy concern during data sharing.

III. Experimental Design

In this section, we describe the design of our experiment, including the studied datasets, the proposal of the federated learning model, and the approach for each research question.

A. Studied Datasets

In this study, we use the datasets provided in the prior CPDP study [8], which are based on project-specific features and the presence or absence of defects, covering both open-source and closed-source projects. A summary of studied datasets is shown in Table I. Datasets are grouped according to the language, use of the project, and features. As shown in the table, the column Projects shows the number of projects per group. The column All Versions represents the total number of all versions of the project, as some of the projects have more than one version. The column Features represents the number of studied features in the group. In the column Project Type, OSS represents open-source software while CSS denotes closed-source software. In this study, a total of 25 projects (i.e., twenty open-source projects and five closed-source projects) with 59 versions from four groups of datasets are selected.

TABLE I
A summary of studied datasets

Group	Projects	All Versions	Features	Project Type	Language
AEEEM	5	5	31	OSS	Java
METRICSREPO	12	46	24	OSS	Java
RELINK	3	3	26	OSS	Java
SOFTLAB	5	5	29	CSS	C
Total	25	59			

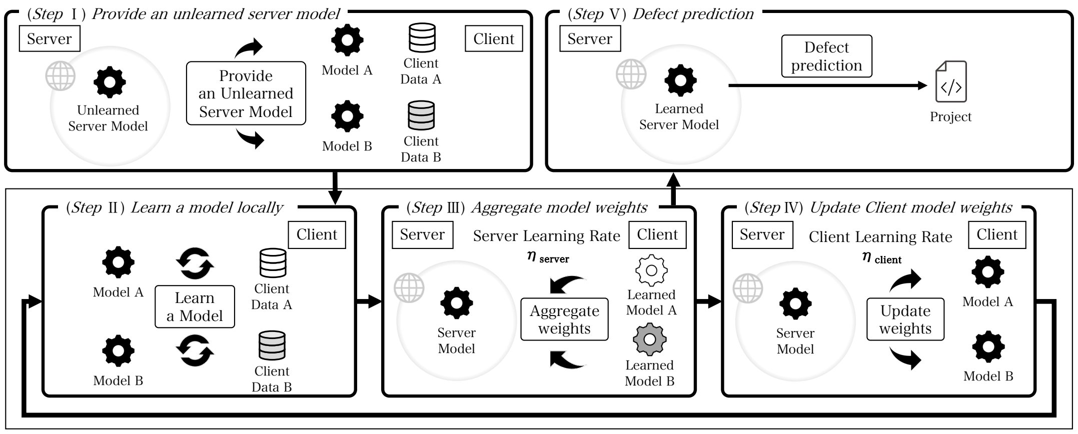


Fig. 2. The overview of the proposed federated learning based CPDP model (FLR) between a Server and Clients

B. Federated Learning Based Cross-Project Defect Prediction

In this study, we propose a federated learning based CPDP model, called *FLR*, for the aim of privacy preservation. To implement federated learning, we leverage Tensorflow Federated[1] as our model framework. Tensorflow Federated is a framework that allows the simulation of federated learning for neural network models in a local environment. Logistic regression can be reproduced by neural networks because of its simple structure. Therefore, we reproduce Logistic Regression (LR) in a single-node neural network model using the Sigmoid function and incorporate it into federated learning. Figure 2 illustrates the overview of federated learning based CPDP flow. Below, we describe how the proposed CPDP model works step by step:

- *(Step I): Provide an unlearned Server model.* In the first step, the Server provides an unlearned Server model to all participating Clients (Clients' models). In our work, Server model refers to the Logistic Regression model and a Client denotes each version of a project in the studied datasets.
- *(Step II): Learn a model locally.* After obtaining the unlearned Server model, each participating Client then uses the own data, trains the obtained model locally, and updates the model weights.
- *(Step III): Aggregate model weights to Server.* In this step, the Client shares only the weights of the updated model with the Server model. A parameter called Server learning rate η_{server} is used to determine the learning rate of weights to the Server model. The learning rate is a value that measures how much the parameters of the model are updated. The Server performs an averaging process on the weights that are aggregated from the Client's model and then updates the parameter information in the Server model.
- *(Step IV): Update Client model weights.* The Server again distributes the parameter information of the updated Server model to the Client model and updates the weights of the Client model. A parameter called the Client learning rate η_{client} is used to determine the learning rate of weights to the Client's model. This ensures that each Client's model has the weights obtained from the Server model and that all Clients' models are updated to have a common weight.
- *(Step V): Defect prediction.* The updating and sharing of the parameter information of the models owned by the Server and the Client from *Step II* to *Step IV* is considered as *one round*. Afterward, a specified number of rounds are repeated to complete the full learning of the Server model. Finally, the Server model that has been fully learned is used to predict defects in the project. For the parameters of federated learning used in the experiment, the number of rounds was set to 100 to check the transition of the binary cross-entropy which is a loss function.

[1]https://www.tensorflow.org/federated

C. Performance of FLR (RQ1)

To address RQ1: How well do federated learning CPDP models perform compared to traditional privacy preserved CPDP methods?, we compare the performance of our proposed federated learning based CPDP (*FLR*) against the (i) privacy-preserving methods (i.e., LACE2) and (ii) conventional CPDP methods (i.e., supervised and unsupervised learning). Below, we describe the studied baseline methods (i.e., seven in total), along with their selection rationales in detail:

- Supervised Learning (SL): Three conventional kinds of supervised learning models are elected: Logistic Regression (LR), Random Forest (RF), and K-Nearest Neighbor (KNN). For the choice of LR, since we propose a Federated Logistic Regression (*FLR*), we would like to compare the proposed method against the one without federated learning algorithm fairly. RF is considered, since it is regarded as one of the best performing models in the context of supervised learning based CPDP [34]. KNN is selected, because it is the model that was applied in the LACE2 algorithm [11].

- LACE2: In this study, we focus on LACE2, which is a privacy algorithm for multi-party data sharing proposed by Peters et al. [11]. We choose this algorithm, since it performs better than the prior privacy approach, reaching higher accuracy. To construct the baseline models, we apply LACE2 to the three studied supervised learning methods (i.e., LR, RF, KNN), respectively.
- Unsupervised Learning (UL): Manualdown (MD) [8] is selected as our comparable UL model. ManualDown considers a larger module as more defect-prone. Moreover, the prior work [8] recommended that Manualdown should be taken as the baseline model in new CPDP models.

We use the datasets (i.e., 25 projects with 59 versions) described in Section III-A to evaluate the performance of the federated learning based CPDP model (*FLR*) and the elected baseline models. Specifically, for the choice of testing data in these studied models, we regard the latest version of each project as testing data, i.e., 25 versions being identified from 25 projects. Since we focus on cross-project defect prediction, when we use a project (i.e., an identified version) as testing data, we do not use the non-latest versions of this project as training data. Instead, all the versions, including the latest versions, of the remaining projects are treated as training data. The target of the training data is whether the project contains defects (a value of 0 or 1). For the *FLR* model, SL (supervised learning) models, and LACE2 models, we conducted 100 rounds of defect prediction for one testing data by randomly sampling the training data to reduce the data selection bias. We select 80% of the versions of the training data as the sampled training data for each round. For the UL (unsupervised learning) models, taking into account its nature, one round is conducted for one testing data regardless of the training data. The optimal parameters for each model are calculated by performing a grid search. We used GridSearchCV function provided by the sklearn packages [35].

Evaluation Metrics. We evaluate the proposed *FLR* model and the seven baseline CPDP models in terms of two viewpoints: *non-effort-aware performance measures (NPMs)* and *effort-aware performance measures (EPMs).* We use four NPMs and one EPM, and the details are described below.

Non effort-aware performance measure (NPMs). NPMs are commonly used in defect prediction studies [36]. They measure how accurately defect prediction models predict defective entities (e.g., files). We use the following two kinds of NPMs: threshold-dependent measures (precision, recall, and F1), and threshold-independent measures (Area Under the receiver operating characteristic Curve, AUC). For the threshold-dependent measures, similar to the prior work [37], we set a commonly-used probability threshold of 0.5. As suggested by Tantithamthavorn and Hassan [38], threshold-independent measures should be considered since threshold-dependent measures can lead to different and conflicting conclusions.

Effort-aware performance measure (EPMs). Xia et al. [39] reported that NPMs are difficult to provide enough information to help practitioners to evaluate CPDP models due to limited testing resources. Hence, we also examine EPMs [37, 40, 36, 1]. EPMs can consider to what extent resources are required to repair identified defects. Specifically, we use an EPM: Cost-Effort@L, which measures the number of identified defective entities (i.e., defective files) when developers inspect L lines of code. We include the following three L variants: 20% (CostEffort@20%), 1,000 lines of codes (CostEffort@1000), and 2,000 lines of codes (CostEffort@2000).

To further statistically compare the evaluation measure (i.e., AUC) across the studied models, we use the Scott-Knott ESD test [41] as our statistical test. The Scott-Knott ESD test ranks the distributions based on not only statistically significant differences but also Cohen's d effect size. Note that the Scott-Knott ESD test is performed on the seven models (i.e., *FLR*, three SL based models, and three models employing LACE2), since for UL based models, only one round is conducted for one testing data.

D. Feature Interpretation (RQ2)

To address RQ2: What features affect the CPDP model using federated learning?, we adopt a common way to interpret defect prediction models by using feature importance, which measures to what degree each feature contributes to the model's prediction. We use the AEEEM, METRICSREPO, and RELINK datasets. Since this study uses a neural framework as described in Section III-B, we interpret the model by using permutation analysis [42], which is one of the analysis methods that does not use the internal structure of the model. In permutation analysis, a model is learned by reordering completely at random certain features of the training data. Therefore, the reordered features no longer have the ability to explain the target. In other words, if the prediction performance of a reordered feature is significantly worse than the prediction performance of the original model, it can be assumed that this feature affects the prediction performance.

As a pretreatment for the experiment, we first reduce the features (explanatory variables) that share collinearity by conducting correlation and redundancy analysis. Highly correlated explanatory variables can interfere with each other when examining the significance, which would potentially lead to spurious conclusions. To do so, similar to the work of McIntosh et al. [43], we use the Spearman rank correlation (ρ) to assess the correlation between each pair of studied features. We remove the pairs of features that have an absolute Spearman correlation coefficient of above or equal to 0.7 [44]. To assure that studied features provide a unique signal, the redun function [45] of the rms package is applied to further remove variables with R^2 above the threshold value of 0.9. After performing the above two feature reduction methods, permutation analysis is then performed on the remaining features to investigate the performance difference in terms of AUC evaluation metric.

Similar to RQ1, the Scott Knott ESD Test is adopted to statistically examine the performance difference (i.e., AUC) for each remaining feature, and the features are statistically grouped into the ranks regarding performance.

IV. Experimental Results

In this section, we present the results of our proposed two research questions.

A. How well do federated learning CPDP models perform compared to traditional privacy preserved CPDP methods?

Table II presents the results of the prediction performance of *FLR* and the seven baseline models for each group of studied datasets. The column Measure represents the diverse evaluation metrics. For all chosen metrics, the larger values are, the better the model performs. The column Loc refers to the average lines of code for the projects in a group. Figure 3 shows the results of the Scott-Knott ESD test for the seven studied models, excluding the unsupervised learning based model. The rank group assignment by the Scott-Knott ESD test explains that the higher the rank is, the better the model is assigned to the group with the best prediction performance. Specifically, Figure 3 shows the cumulative number of times a model is assigned to each rank when the model is grouped by AUC based on evaluation within each of the 25 testing data. We now discuss the two main results below:

FLR model does not perform as well as conventional methods (SL and UL), but it is comparable with lower false alarm. As shown in Table II, we observe that *FLR* model cannot achieve the high F1 scores as the conventional CPDP methods do. For instance, F1 scores of *FLR* model range from 0.175 to 0.378 for the studied four groups, however, the F1 scores of LR model (supervised learning methods) and Manualdown model (MD, unsupervised learning methods) vary from 0.353 to 0.368 and from 0.389 to 0.477, respectively. Furthermore, a closer inspection of Scott-Knott ESD test shown in Figure 3, we see that the *FLR* model does not belong to a higher rank group than the LR model in any of the evaluations upon the four groups. This may be due to the fact that the LR model can use the original data as training data, whereas the *FLR* model uses a model that aggregates the parameters of the model trained by the Client, and thus the data itself is kept confidential.

On the other hand, we find that our proposed *FLR* is still comparable to these conventional CPDP methods in terms of other evaluation metrics. For example, *FLR* model results in a better precision across the baseline models. Specifically, the precision scores of *FLR* model are from 0.491 to 0.714 for the four groups, while the precision scores of SL models (i.e., LR, RF, and KNN) and UL model (i.e., MD) are from 0.316 to 0.611 and from 0.250 to 0.592, separately. The precision indicates how accurately the model identifies defective entities; the higher the precision, the fewer false alarms. Hence, these results suggest that FLR model yields stronger privacy but also provides a less number of false alarms. Moreover, in terms of CostEffort@20%, we observe that *FLR* model shows better evaluation than the UL model, scores being from 0.077 to 0.154. With regards to the Scott-Knott ESD test depicted in Figure 3, the *FLR* model is assigned to the first group 5 times (20 percent) and the second group 12 times (48 percent) for the 25 testing data, suggesting that it also has some comparable prediction potential against the conventional CPDP methods.

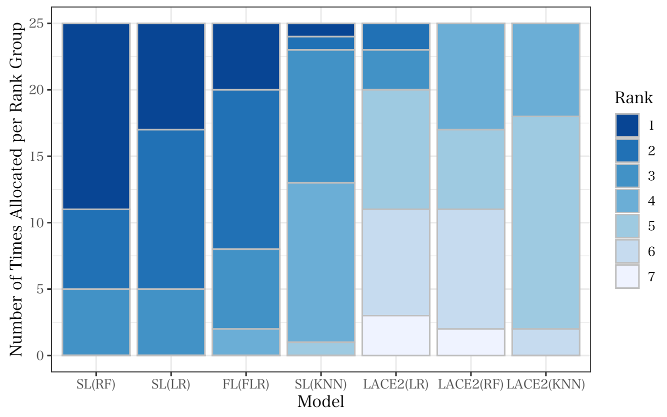


Fig. 3. Distribution of Scott Knott ESD Test rank groups for AUC of the model

FLR model relatively outperforms the existing privacy-preserving methods. As we can see from Table II, *FLR* model yields higher precision and AUC scores greatly than all the models that apply LACE2 algorithm. For instance, the AUC scores of *FLR* model range from 0.684 to 0.770, while the AUC scores of LACE2 ones range from 0.456 to 0.679. Meanwhile, the precision scores of *FLR* model vary from 0.491 to 0.714, however, the AUC scores of the models that apply LACE2 are from 0.134 to 0.296. In addition, the Scott-Knott ESD test results in Figure 3 show that the *FLR* model belongs to a higher rank group than any of the LACE2 models in all testing data across four groups. These findings suggest that *FLR* model relatively performs better than the existing privacy-preserving methods from the practitioner view, though lower recall scores.

RQ1 Summary

Our evaluation results show that the *FLR* model achieves higher prediction performance (i.e., precision and AUC) than the existing privacy-preserving methods (i.e., LACE2). When compared against the conventional methods, the *FLR* model does not perform as well as the conventional methods, but shows comparable prediction potential (i.e., higher precision). Moreover, the statistical tests confirmed that the *FLR* model is likely to be assigned to the higher rank group in prediction performance (68% of the testing data).

B. What features affect the CPDP model using federated learning?

Table III presents the top-3 ranking features in the Scott-Knott ESD test for the evaluated datasets. The column Rank represents the ranks by the Scott-Knott ESD test; the column Feature and the columns Description denotes the feature and

TABLE II
MEDIAN AUC OF MODELS PER DATASET GROUP
(HIGHLIGHTS: FLR MODEL HAS HIGHER AUC AND PRECISION THAN THE THREE LACE2 MODELS)

Type	Measure	Group	Loc	FL FLR	SL LR	SL RF	SL KNN	UL MD	LACE2 LR	LACE2 RF	LACE2 KNN
NPM	AUC	AEEEM	146952	0.684	0.718	0.739	0.632	0.717	0.565	0.543	0.509
		METRICSREPO	110383	0.679	0.709	0.697	0.609	0.693	0.558	0.542	0.532
		RELINK	41110	0.712	0.740	0.679	0.611	0.762	0.542	0.542	0.534
		SOFTLAB	2535	0.770	0.796	0.775	0.632	0.788	0.554	0.542	0.540
	Precision	AEEEM	146952	0.680	0.556	0.611	0.316	0.295	0.134	0.137	0.136
		METRICSREPO	110383	0.620	0.603	0.567	0.344	0.471	0.245	0.199	0.208
		RELINK	41110	0.714	0.611	0.589	0.393	0.592	0.279	0.275	0.273
		SOFTLAB	2535	0.491	0.603	0.611	0.400	0.250	0.270	0.225	0.296
	Recall	AEEEM	146952	0.108	0.172	0.172	0.172	0.733	0.941	0.832	0.980
		METRICSREPO	110383	0.257	0.172	0.193	0.299	0.650	0.300	0.622	0.616
		RELINK	41110	0.290	0.182	0.218	0.332	0.745	0.330	0.616	0.633
		SOFTLAB	2535	0.113	0.156	0.222	0.333	0.852	0.350	0.596	0.616
	F1	AEEEM	146952	0.175	0.353	0.318	0.222	0.409	0.235	0.235	0.239
		METRICSREPO	110383	0.266	0.353	0.323	0.340	0.389	0.270	0.302	0.311
		RELINK	41110	0.378	0.368	0.361	0.359	0.477	0.302	0.381	0.382
		SOFTLAB	2535	0.179	0.349	0.379	0.364	0.408	0.305	0.327	0.400
EPM	CostEffort@20%	AEEEM	146952	0.130	0.164	0.107	0.098	0.040	0.298	0.127	0.206
		METRICSREPO	110383	0.077	0.083	0.092	0.128	0.027	0.205	0.151	0.182
		RELINK	41110	0.085	0.080	0.089	0.132	0.053	0.210	0.150	0.178
		SOFTLAB	2535	0.154	0.106	0.111	0.136	0.125	0.222	0.158	0.175
	CostEffort@1000	AEEEM	146952	0.000	0.000	0.000	0.005	0.000	0.012	0.000	0.005
		METRICSREPO	110383	0.000	0.000	0.000	0.004	0.000	0.005	0.000	0.005
		RELINK	41110	0.008	0.000	0.000	0.004	0.000	0.005	0.000	0.005
		SOFTLAB	2535	0.375	0.000	0.000	0.005	0.375	0.021	0.000	0.012
	CostEffort@2000	AEEEM	146952	0.008	0.008	0.004	0.010	0.000	0.016	0.005	0.010
		METRICSREPO	110383	0.000	0.005	0.000	0.007	0.000	0.012	0.005	0.010
		RELINK	41110	0.025	0.005	0.000	0.008	0.017	0.014	0.006	0.010
		SOFTLAB	2535	1.000	0.008	0.001	0.012	1.000	0.042	0.021	0.023

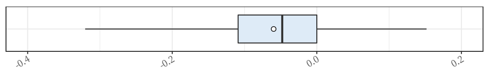
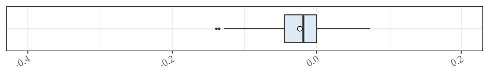
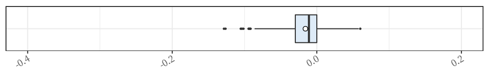
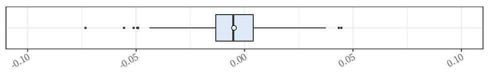
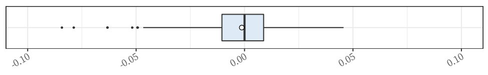
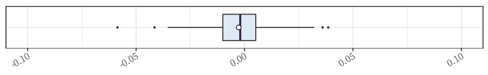
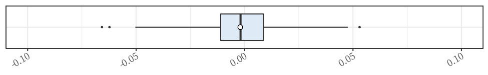
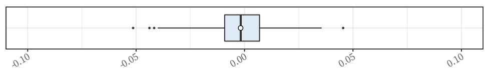
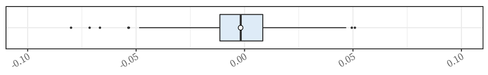
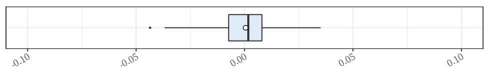
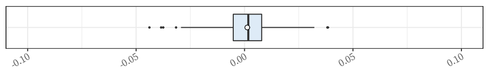
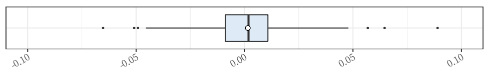
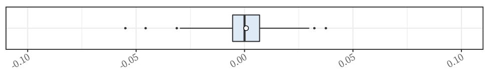
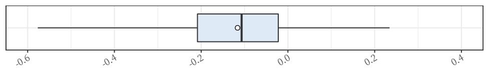
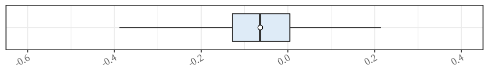
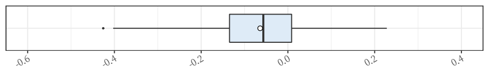
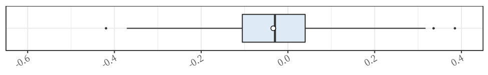
its description, respectively; the column Boxplot refers to the boxplots of AUC differences between the original AUC values in RQ1 and the ones in the permutation analysis. Below we now discuss the important features of the three datasets.

AEEEM. Nine features survived from 31 original features after the correlation and redundancy analysis within this group. The surviving features are concerning the Chidamber and Kemerer (CK) features [48] and the object-oriented features [46]. As shown in Table III, *fanOut* (the number of other classes referred by the class) has a relatively larger effect on the *FLR* model. *fanOut* is a metric that is highly correlated with *cbo* (coupling between object classes). Specifically, *fanOut* achieves the first rank, and its performance difference is visibly greater than the *fanIn* and the *numberOfPublicMethods*. This result suggests that the coupling-related features are more likely to affect the proposed *FLR* model. On the other hand, we also notice that *numberOfPublicMethods*, which is related to the scale category, also shows an effect in the model performance (rank 3).

METRICSREPO. Twelve features remained out of 24 original features, which belong to five categories (i.e., complexity, coupling, cohesion, inheritance, and scale) by Yao et al. [47]. Compared to the other datasets, we observe that the number of features in the top 3 ranks is larger, implying that the feature importance is relatively equally distributed across each feature within the METRICSREPO dataset. The top 1 rank, however, only includes one feature *npm* (number of public methods) as shown in Table III. The correlation analysis showed that npm is strongly correlated with *wmc* (weighted methods per class). Both features are classified into the scale category as per the prior work. Such a result suggests that scale-related features may greatly contribute to the fit of our *FLR* model.

RELINK. Four features (*AvgEssential*, *AvgLineBlank*, *RatioCommentToCode*, *SumEsssential*) survived from 26 features after the correlation and redundancy analysis. During the analysis, for instance, we found that the *AvgEssential* is strongly correlated with the cyclomatic complexity features, such as the mean value of cyclomatic complexity. While *SumEssential* is highly correlated to the features regarding the code itself, such as the number of lines of code or lines of code executed. Table III shows that the top 1 rank is *SumEssential* (sum of essential complexity of all nested functions or methods). This observation indicates that the feature regarding the code itself (i.e., complexity) is relatively more important than the other features in the *FLR* model. Meanwhile, similar to the prior two evaluated datasets, we find that scale-related features (i.e., *AvgLineBlank*, *RatioCommentToCode*) also play a role, ranked as 2 and 3, respectively.

TABLE III
DISTRIBUTION OF SCOTT KNOTT ESD TEST RANK GROUPS FOR AUC WITH PER GROUP (TOP3)

Group	Rank	Feature	Description	Boxplot (Difference in AUC with and without permutation analysis)
AEEEM[46]	1	fanOut	Number of other classes referred by the class	
	2	fanIn	Number of other classes that refer to the class	
	3	numberOfPublicMethods	Number of public methods	
METRICSREPO[47]	1	npm (Number of Public Methods)	Number of public methods	
	2	avg_cc (Average McCabe)	Average McCabe's Cyclomatic Complexity values of methods in the same class	
	2	ca (Afferent Couplings)	How many other classes use the specific class	
	2	ce (Efferent Couplings)	How many other classes is used by the specific class	
	2	dit (Depth of Inheritance Tree)	Provides the position of the class in the inheritance tree	
	2	ic (Inheritance Coupling)	Number of parent classes to which a given class is coupled	
	3	amc (Average Method Complexity)	Average method complexity (e.g., using number of java byte codes)	
	3	lcom (Lack of Cohesion in Methods)	Number of pairs of methods that do not share a reference to an instance variable	
	3	lcom3 (Lack of Cohesion in Methods, different from lcom)	If m, a are the number of methods, attributes in a class number and μ(a) is the number of methods accessing an attribute, then $lcom3 = ((\frac{1}{a}\sum_{j}^{a}\mu(a_j)) - m)/(1-m)$	
	3	noc (Number of Children)	Measures the number of immediate descendants of the class	
RELINK	1	SumEssential	Sum of Essential complexity of all nested functions or methods	
	2	AvgEssential	Average Essential complexity for all nested functions or methods	
	2	AvgLineBlank	Average number of blank for all nested functions or methods	
	3	RatioCommentToCode	Ratio of comment lines to code lines	

RQ2 Summary

The results of the feature interpretation analysis show that on the one hand, the most important feature differs from the evaluated datasets. On the other hand, we find that scale-related features commonly have an effect on the prediction performance of the *FLR* model in terms of the top-3 features.

V. Discussion

In this section, we further discuss the effect of the learning rate and the client participation on the prediction performance.

A. Does the Server/Client learning rate improve prediction performance?

Objective. In the *FLR* model approach, model weights are exchanged multiple times between the Server and Client models. Two parameters used in this process are η_{server} and η_{client} as we describe in Section III-B. η_{server} applies the weights updated at the client to the global model at the server. η_{client} is used to compute weight updates for the respective local models at the client. The two learning rates are both expressed as a value between 0 and 1, and are calculated as follows when updating parameters, similar to the gradient descent method in machine learning. In the following formula, w denotes the parameter, η is the Server or Client learning rate, and g represents the gradient of the cost function.

$$w = w - \eta \times g \tag{1}$$

These two learning rates are significant parameters that lead to how much of the model parameters are received. Thus, we would like to examine the relationship between the learning rates and the model performance.

Approach. To address this, we investigate the performance change of *FLR* models when the parameters η_{client} and η_{server} for the Client and Server models in the *FLR* model are combined at values of [0.01, 0.05, 0.1, 0.5, 1.0], respectively. Four datasets are used, and all eight different models are trained and tested in the same manner as RQ1. AUC is used as the evaluation metric.

Results. Figure 4 (a) presents the AUC change of the *FLR* model where η_{client} is fixed and η_{server} is varied and Figure 4 (b) shows the AUC change of the *FLR* model where the value of η_{server} is fixed and η_{client} is changed. The results show that for all η_{server}, there is a large difference when η_{server} is set between 0.01 and 0.05. Conversely, significant performance improvement is not observed between 0.10 and 1.0. As shown in Figure 4 (b), when the Server learning rate η_{server} is fixed and the Client learning rate η_{client} is varied, the performance difference between η_{client} of 0.01 and 0.05 is also larger than the other cases (i.e., between 0.10 and 1.0).

These results suggest that for both η_{server} and η_{client}, when the value is up to 0.05, it has a significant impact on CPDP performance improvement, while the values larger than 0.10 do not show significant performance improvement.

Observation I

For both η_{server} and η_{client}, the learning rates for the exchange of parameters between the Server and the Client, when the value is up to 0.05, there is a large impact on the performance improvement of the CPDP. However, when the value is larger than 0.10, there is no significant performance improvement.

B. Does predictive performance improve as client participation increases?

Objective. One of the important characteristics of federated learning approach is the presence of Clients. Knowing to what extent the presence of Clients affects prediction performance is one indicator for estimating the number of participants and the required data to inject federated learning into CPDP. Hence, we would like to further examine the relationship between the number of Clients and the forecasting performance.

Approach. To shed light, we use the data from the METRICSREPO group, which contains the largest number of projects and versions (i.e., 12 projects with 46 versions). Therefore, this dataset is an appropriate candidate used to see the impact of changes in the number of Clients over a wide range. The *FLR* model is then trained and tested similar to RQ1, by measuring the change in performance when the number of Clients is varied within the list of [1, 10, 20, 30, 40]. AUC is used as the evaluation metric.

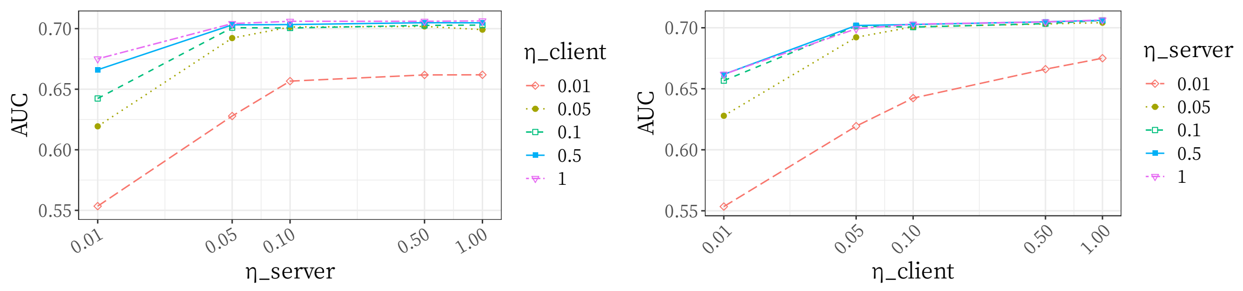
Results. Figure 5 shows the AUC change in the *FLR* model when the number of Clients is increased for each project under the METRICSREPO group. With the number of Clients changing from 1 to 10, the model performance gets relatively improved greatly for all projects. When the number of Clients is larger than 10, on the one hand, the Ivy, PBeans, and Synapse projects show increasing performance improvements slightly. On the other hand, the performance of the other projects does not visibly get improved as the number of Clients increases, and the performance improvement tends to be flat. This could be implied that the weights of the Server model aggregated from the Client models become closer to the average as the number of projects in the same group increases with the number of Clients since data from the same group is used for feature unification.

Observation II

Results show that increasing the number of Clients tends to improve performance when the number of Clients is up to 10. However, the performance does not get great improvement when the number of Clients is greater than 10.

VI. Threats to Validity

In this section, we disclose the threats to validity.



(a) AUC for each η_{client} when η_{server} is Fixed **(b) AUC for each η_{server} when η_{client} is Fixed**

Fig. 4. AUC variation by η_{server} and η_{client}

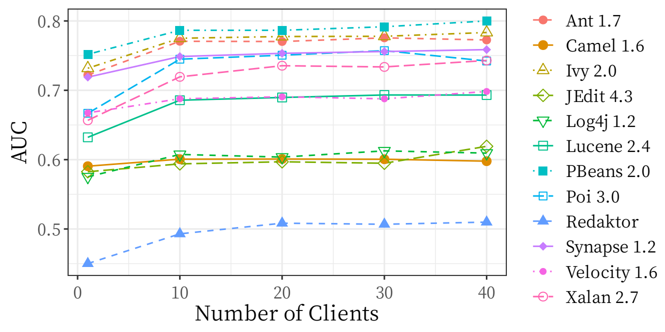


Fig. 5. AUC by Number of Clients in METRICS REPO Projects

a) External Validity: This validity is with regard to the generalization ability of the results. In this study, we evaluate the performance proposed *FLR* model against the existing CPDP models, using 25 projects. The threat may occur due to the number or the variety of the selected projects. However, we are confident that these projects are representative enough to provide insights, as they are widely studied in the prior work [8]. Moreover, in RQ2, we excluded the SOFTLAB dataset to investigate what features would play a role in *FLR* model, due to the time limitation. The results may also not be generalized to other projects. While our goal is to shed light on the effect of important features. Nevertheless, future research should be extended to take more projects into account and see whether or not the effect commonly exists.

b) Internal Validity: It refers to the approximate truth about inferences. The threat could occur during the reproduction of the existing models. In this study, we implement the seven CPDP methods, including supervised learning, LACE2, and unsupervised learning. The choice of the parameter may affect the accurate reproduction, however, to avoid this threat, we carefully inspected the literature and reproduced the method step by step. In addition, to measure the performance, we adopt commonly used evaluation metrics, e.g., precision, AUC, and CostEffort. The evaluation conclusion may differ from the other evaluation metrics.

c) Construct Validity: This validity is related to the degree to which our measurements capture. To implement the *FLR* model, we apply the Tensorflow Federated. Tensorflow Federated is a framework that enables execution in a simulation environment, but not in an actual data-distributed environment. Thus, the constructed threat would exist due to the different environment. Future research should be extended to be conducted in a data-distributed environment.

VII. Conclusion & Future Work

We propose a CPDP model in this paper, coined as *FLR*, that incorporates logistic regression to federated learning, to address the challenge of privacy preservation. We then perform an empirical study using 25 projects to evaluate their effectiveness and interpret the important features. In terms of effectiveness, our results show that the *FLR* model achieves higher precision and AUC than the existing models that apply the privacy preservation algorithm LACE2. In terms of feature interpretation, the results demonstrate that although the most important feature differs from the evaluated datasets, the scale-related features are likely to have a common effect on the prediction performance of the *FLR* model.

Our work confirms the promise of CPDP models that employ federated learning. At the same time, we open up several future research directions. For instance, the results of the RQ2 experiments are based on open-source datasets due to limited time, thus, closed-source datasets (e.g., SOFTLAB) should be taken into account to increase the generality of the results. Meanwhile, a further control study is supposed to be conducted to compare against those CPDP models without federated learning to confirm the effect of federated learning on the feature values. Regarding the model selection used in federated learning, our study only confirms the performance of the Logistic Regression model, but other models used in CPDP are also suggested to be incorporated into federated learning and investigated.

Acknowledgement

This research was partially supported by JSPS KAKENHI Japan (Grant Numbers: JP18H04097, JP20H04167, JP21H04877, JP22K17874, JP22K18630) and JSPS International Joint Research Program with SNSF (Project "SENSOR": JPJSJRP20191502).

References

[1] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, "A large-scale empirical study of just-in-time quality assurance," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 757–773, 2013.

[2] Z. Wan, X. Xia, A. E. Hassan, D. Lo, J. Yin, and X. Yang, "Perceptions, expectations, and challenges in defect prediction," *IEEE Transactions on Software Engineering*, vol. 46, no. 11, pp. 1241–1266, 2018.

[3] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process," in *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, 2009, p. 91–100.

[4] S. Hosseini, B. Turhan, and D. Gunarathna, "A systematic literature review and meta-analysis on cross project defect prediction," *IEEE Transactions on Software Engineering*, vol. 45, no. 2, pp. 111–147, 2017.

[5] Q. Zou, L. Lu, Z. Yang, X. Gu, and S. Qiu, "Joint feature representation learning and progressive distribution matching for cross-project defect prediction," *Information and Software Technology*, vol. 137, p. 106588, 2021.

[6] G. Boetticher, "The promise repository of empirical software engineering data," *http://promisedata.org/repository*, 2007.

[7] Z. He, F. Peters, T. Menzies, and Y. Yang, "Learning from open-source projects: An empirical study on defect prediction," in *Proceedings of the 7th IEEE International Symposium on Empirical Software Engineering and Measurement*, 2013, pp. 45–54.

[8] Y. Zhou, Y. Yang, H. Lu, L. Chen, Y. Li, Y. Zhao, J. Qian, and B. Xu, "How far we have progressed in the journey? an examination of cross-project defect prediction," *ACM Transactions on Software Engineering and Methodology*, vol. 27, no. 1, pp. 1–51, 2018.

[9] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, "Heterogeneous cross-company defect prediction by unified metric representation and cca-based transfer learning," in *Proceedings of the 10th joint meeting on foundations of software engineering*, 2015, pp. 496–507.

[10] F. Peters and T. Menzies, "Privacy and utility for defect prediction: Experiments with morph," in *Proceedings of the 34th International Conference on Software Engineering*, 2012, p. 189–199.

[11] F. Peters, T. Menzies, and L. Layman, "Lace2: Better privacy-preserving data sharing for cross project defect prediction," in *Proceedings of the 37th IEEE International Conference on Software Engineering*, vol. 1, 2015, pp. 801–811.

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

[13] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2019.

[14] 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*, 2019, pp. 1–11.

[15] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2019.

[16] Replication package. [Online]. Available: https://github.com/posl/SANER2023_CPDPwithFL

[17] N. Li, M. Shepperd, and Y. Guo, "A systematic review of unsupervised learning techniques for software defect prediction," *Information and Software Technology*, vol. 122, p. 106287, 2020.

[18] G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella, "Multi-objective cross-project defect prediction," in *Proceedings of the 6th International Conference on Software Testing, Verification and Validation*, 2013, pp. 252–261.

[19] A. Panichella, R. Oliveto, and A. De Lucia, "Cross-project defect prediction models: L'union fait la force," in *Proceedings of the 2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering*, 2014, pp. 164–173.

[20] Y. Zhang, D. Lo, X. Xia, and J. Sun, "An empirical study of classifier combination for cross-project defect prediction," in *Proceedings of the 39th Annual computer software and applications conference*, vol. 2, 2015, pp. 264–269.

[21] L. Gong, S. Jiang, L. Bo, L. Jiang, and J. Qian, "A novel class-imbalance learning approach for both within-project and cross-project defect prediction," *IEEE Transactions on Reliability*, vol. 69, no. 1, pp. 40–54, 2020.

[22] J. Nam and S. Kim, "Clami: Defect prediction on unlabeled datasets (t)," in *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*, 2015, pp. 452–463.

[23] F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan, "Cross-project defect prediction using a connectivity-based unsupervised classifier," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 309–320.

[24] Z. Li, X.-Y. Jing, and X. Zhu, "Progress on approaches to software defect prediction," *IET Software*, vol. 12, no. 3, pp. 161–175, 2018.

[25] E. J. Weyuker, T. J. Ostrand, and R. M. Bell, "Do too many cooks spoil the broth? using the number of developers to enhance defect prediction models," *Empirical*

Software Engineering, vol. 13, no. 5, pp. 539–559, 2008.

[26] F. Peters, T. Menzies, L. Gong, and H. Zhang, "Balancing privacy and utility in cross-company defect prediction," *IEEE Transactions on Software Engineering*, vol. 39, pp. 1054–1068, 2013.

[27] Y. Fan, C. Lv, X. Zhang, G. Zhou, and Y. Zhou, "The utility challenge of privacy-preserving data-sharing in cross-company defect prediction: An empirical study of the cliff&morph algorithm," in *Proceedings of the 33rd International Conference on Software Maintenance and Evolution*, 2017, pp. 80–90.

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

[29] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[30] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[31] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[32] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[33] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang, "Fedvision: An online visual object detection platform powered by federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 08, 2020, pp. 13 172–13 179.

[34] L. Goel, M. Sharma, S. K. Khatri, and D. Damodaran, "Prediction of cross project defects using ensemble based multinomial classifier," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 7, pp. 1–14, 2020.

[35] Gridsearchcv. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[36] C. Ni, X. Xia, D. Lo, X. Chen, and Q. Gu, "Revisiting supervised and unsupervised methods for effort-aware cross-project defect prediction," *IEEE Transactions on Software Engineering*, vol. 48, no. 3, pp. 786–802, 2022.

[37] M. Kondo, Y. Kashiwa, Y. Kamei, and O. Mizuno, "An empirical study of issue-link algorithms: which issue-link algorithms should we use?" *Empirical Software Engineering*, vol. 27, pp. 1–50, 2022.

[38] C. Tantithamthavorn and A. E. Hassan, "An experience report on defect modelling in practice: Pitfalls and challenges," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, 2018, p. 286–295.

[39] X. Xia, D. Lo, S. J. Pan, N. Nagappan, and X. Wang, "Hydra: Massively compositional model for cross-project defect prediction," *IEEE Transactions on Software Engineering*, vol. 42, no. 10, pp. 977–998, 2016.

[40] Q. Huang, X. Xia, and D. Lo, "Supervised vs unsupervised models: A holistic look at effort-aware just-in-time defect prediction," in *Proceedings of the 33rd International Conference on Software Maintenance and Evolution*, 2017, pp. 159–170.

[41] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 1–18, 2016.

[42] G. K. Rajbahadur, S. Wang, G. Ansaldi, Y. Kamei, and A. E. Hassan, "The impact of feature importance methods on the interpretation of defect classifiers," *IEEE Transactions on Software Engineering*, vol. 48, no. 7, pp. 2245–2261, 2021.

[43] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "An empirical study of the impact of modern code review practices on software quality," *Empirical Software Engineering*, vol. 21, p. 2146–2189, 2015.

[44] H. C. Kraemer, G. A. Morgan, N. L. Leech, J. A. Gliner, J. J. Vaske, and R. J. Harmon, "Measures of clinical significance," *Journal of the American Academy of Child & Adolescent Psychiatry*, vol. 42, no. 12, pp. 1524–1529, 2003.

[45] redun. [Online]. Available: https://www.rdocumentation.org/packages/Hmisc/versions/4.7-0/topics/redun

[46] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: A benchmark and an extensive comparison," *Empirical Software Engineering*, vol. 17, pp. 1–47, 2012.

[47] Z. Yao, J. Song, Y. Liu, T. Zhang, and J. Wang, "Research on cross-version software defect prediction based on evolutionary information," *IOP Conference Series: Materials Science and Engineering*, vol. 563, p. 052092, 2019.

[48] S. Chidamber and C. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, 1994.