Hindawi
Scientific Programming
Volume 2022, Article ID 4648468, 16 pages
https://doi.org/10.1155/2022/4648468

Research Article

An Approach to Improving Homogeneous Cross-Project Defect Prediction by Jensen-Shannon Divergence and Relative Density

Jiajia Ren,[1] **Chunyu Peng**,[2] **Shang Zheng**,[2] **Haitao Zou**,[2] **and Shang Gao**[2]

[1]*School of Economics and Management, Jiangsu University of Science and Technology, Zhenjiang 212100, China*
[2]*School of Computer Science, Jiangsu University of Science and Technology, Zhenjiang 212100, China*

Correspondence should be addressed to Shang Zheng; szheng@just.edu.cn

Received 13 June 2022; Revised 8 September 2022; Accepted 6 October 2022; Published 20 October 2022

Academic Editor: Shujuan Jiang

Homogeneous cross-project defect prediction (HCPDP) aims to apply a binary classification model built on source projects to a target project with the same metrics. However, there is still room for improvement in the performance of the existing HCPDP models. This study has proposed a novel approach, including one-to-one and many-to-one predictions. First, we apply the Jensen-Shannon divergence to select the most similar source project automatically. Second, relative density estimation is introduced to choose the suitable instance of the selected source project. Third, one-to-one and many-to-one prediction models are trained by the selected instances. Finally, two benchmark datasets are used to evaluate the proposed approach. Compared to the state-of-the-art methods, the experimental results demonstrated that the proposed approach could improve the prediction performance in the F1-score, AUC, and G-mean metrics and exhibit strong adaptability to the traditional classifiers.

1. Introduction

Software defects are inevitable in the process of software development. Software defects affect user experience and bring economic losses to enterprises. Therefore, it is necessary to help software engineers accurately predict defects in projects. Recent years have seen an increasing trend in software defect prediction (Hall et al. [1]; Rahman et al. [2]; Yang et al. [3]; Ghotra et al. [4], Wen et al. [5]; Tang et al. [6]; Jiarpakdee et al. [7]). Previous research has demonstrated that machine-learning approaches are suitable for software defect prediction (Menzies et al. [8]; Zimmermann and Nagappan [9]; Hassan [10]). The project manager can use the defect prediction model to determine whether a module has defects. However, it is difficult to build an effective prediction model if not enough data are available. An alternative solution is cross-project defect prediction (CPDP) (He et al. [11]; Ma et al. [12]; Canfora et al. [13]; Herbold et al. [14]; Hosseini et al. [15]), which constructs the classifiers on existing projects with sufficient labeled data and predicts the defects of the target project. When the metrics (features) between the source and target projects are the same, CPDP is called homogeneous cross-project defect prediction (HCPDP).

Watanabe et al. [16] introduced the metric compensation method, which revises the metrics of the target project based on the metrics of the source project. Turhan et al. [17] calculated the Euclidian distance of each instance and selected the nearest instances to complete defect prediction, but it was one-phase filtering. Herbold [18] calculated the feature vector of each metric and applied two strategies to select suitable source projects. Fukushima et al. [19] proposed an approach to consider the similarity between the source and target projects by calculating the Spearman values. Panichella et al. [20] applied ensemble learning to improve CPDP performance. Subsequently, two-phase filtering methods (He et al. [21]; He et al. [22]) were proposed to improve the work of Turhan. These pieces of work selected the source project and applied the filtering method to construct the prediction models. Due to the development of transfer learning, Nam et al. [23] applied transfer component analysis to search the feature mapping space for the source and target projects and constructed the prediction models. Liu et al. [24] proposed a two-phase transfer

learning method, which used a source project estimator to select similar projects by measuring the Euclidian distance.

Most prior studies focused on one-to-one defect prediction, and their findings have proved that selecting a suitable project is the primary solution for HCPDP. However, there is still room for improvement in the selection process, and many-to-one defect prediction is necessary for a natural environment. Therefore, this paper expands the Jensen-Shannon (JS) divergence (Zheng et al. [25]) to many-to-one defect prediction and applies a relative density strategy together to select the training instances. Unlike choosing the features in prior studies, JS divergence can keep all the features and calculate the similarity of two probability distributions for two projects directly. Moreover, prior work (Turhan et al. [17]; He et al. [22]; Nam et al. [23]) considered using distance or probability density to select the training instance, but it is difficult to choose when the metric dimension is high. This paper presents an alternative to solving this problem. Although it is not easy to exactly obtain the probability density of each instance, it is feasible to extract the proportional relation between the probability densities of any two instances. Therefore, the relative density is proposed to reflect the proportional relations for selecting instances in the projects. Finally, a novel weighting strategy in many-to-one defect prediction is designed. The experimental results show that the proposed approach can improve HCPD performance and be adaptable to different machine-learning algorithms.

The main contributions of this study are summarized as follows:

(1) An approach for HCPDP based on the JS divergence with relative density is designed.

(2) An ensemble weighting strategy based on JS divergence is proposed to build a many-to-one prediction model.

(3) The proposed approach is not subject to change in the learning algorithms.

To evaluate the proposed approach, two benchmark datasets from NASA (Shepperd et al. [26]) and PROMISE (Jureczko and Madeyski [27]) were selected. Compared to the previous studies, one-to-one prediction exhibited average improvements in the F1-score of 13%~119% on NASA and 5%~33% on PROMISE, in the AUC of 8%~31% on NASA and 2%~30% on PROMISE, in the G-mean by 10%~68% on NASA and 3%~56% on PROMISE; many-to-one prediction was averagely improved by 10%~193% of the F1-score on NASA and 15%~63% on PROMISE, 1%~25% of the AUC on NASA and 3%~14% on PROMISE, 2%~75% of the G-mean on NASA and 10%~34% on PROMISE. Moreover, five widely used methods, including Logistic Regression (LR), Naive Bayesian (NB), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Random Forest (RF), were selected to test the proposed approach.

The remainder of this paper is organized as follows: Section 2 presents the preliminary results related to this study. The proposed approach is described in Section 3. Section 4 gives the experiment setup. The experiment results and analysis are shown in Section 5, and Section 6 introduces the threats to validity. Section 7 outlines the related work, and conclusion and future works are provided in Section 8.

2. Preliminary

This section introduces the preliminary result related to the proposed approach, including Jensen–Shannon divergence and the relative density estimation strategy.

2.1. Jensen-Shannon Divergence. Feature selection is often used in CPDP, and filtering, wrapper, and embedded methods are the main methods. The filtering method is very fast, but the selected features may not be useful for the model. The advantage of the wrapper method is that the selected features can improve the effect of the model, but the disadvantage is that the model needs to be trained many times to evaluate the effect of the features. The embedded method considers the advantages of the above two methods, but it is difficult to define a criterion for judging whether a feature is valid or not. Therefore, this paper considered keeping all the features and chose Jensen-Shannon (JS) divergence to measure the similarity of two probability distributions. In probability, JS divergence has the ability to measure the similarity of two distributions, which is based on Kullback–Leibler (KL) divergence (Lamberti and Majtey [28]). Compared with KL divergence, JS divergence has symmetry, which is suitable for cross-project defect prediction. For example, when A is a target project and B is a source project, JS (A||B) can be calculated. However, JS (B||A) is not calculated again because of the symmetry of JS divergence, that is, the two values are the same. Hence, JS divergence can reduce the analysis process of CPDP. Moreover, the range of JS divergence is [0,1], and its value is a constant, which discriminates the similarity of two projects more accurately.

Suppose there are two distributions, P and Q, in which P is the true distribution and Q is the approximate distribution. Then the KL divergence is defined as

$$KL(P\|Q) = \sum_{x\in D} P(x)\log\frac{P(x)}{Q(x)}. \tag{1}$$

However, KL divergence is asymmetric. In other words, $KL(P\|Q) \neq KL(Q\|P)$. Then, the JS divergence is proposed, and it is represented by (2) as follows:

$$JS(P\|Q) = \frac{1}{2}KL\left(P\|\frac{P+Q}{2}\right) + \frac{1}{2}KL\left(Q\|\frac{P+Q}{2}\right). \tag{2}$$

Then MONTE CARLO is used to calculate JS value by (3) as follows:

$$D_{MC}(P\|Q) = \frac{1}{n}\sum_{i=1}^{n}\log p\frac{(x_i)}{q(x_i)} \longrightarrow D(P\|Q). \tag{3}$$

As described in work (Lamberti & Majtey [28]), The smaller the JS divergence, the more similar the source project

is to the target project. In summary, JS divergence is used to avoid the selection randomness of source projects.

2.2. Relative Density Estimation Strategy. After a similar source project is selected, it is necessary to filter the suitable instances as the training data. If all the instances are used, noises and outliers will be drawn. The probability density can be calculated to distinguish significant instances from outliers. However, it is difficult to obtain the exact results when the metric dimension is high. Moreover, it is time-consuming as well.

This subsection presents an alternative to solving this problem. It is not easy to exactly obtain the probability density of each instance, but it is feasible to extract the proportional relation of the probability densities between any two instances. In this study, the relative density reflects the proportional relation. To calculate the relative density, a K-nearest neighbors-based probability density estimation (KNN-PDE) (Fukunaga and Hostetler [29]; Mack and Rosenblatt [30]; Yu et al. [31]; Zheng et al. [32]) alike strategy is adopted. KNN-PDE calculates the Kth nearest neighbor distance of a single instance to measure its probability density distribution. When the number of instances tends to infinity, the results obtained from KNN-PDE converge to the actual probability density distribution.

For each training instance x_i, its Kth nearest neighbors are easy to find, and the distance between them is defined as d_i^K. If d_i^K is larger, x_i will hold a lower density. For noise and outliers, they should appear in the region of low density, so d_i^K can be used to measure the significance of each instance. To obtain a higher value for high-density instances and a lower value for noise and outliers, the reciprocal of d_i^K can be used. The reciprocal of the K-nearest neighbors' distance of each instance is named as its relative density. The relative density between any two samples is inversely proportional to the Kth nearest neighbor distance between them.

$$1/d_i^K 1/d_j^K = \frac{d_j^K}{d_i^K}. \tag{4}$$

According to (4), the selection of K is essential. If it is too small, noise and outliers are challenging to identify, but if it is too large, some small disjunctions will become blurred. The appropriate value for K was discussed in Section 5. After the estimation process was completed, all the instances were sorted, and then they were ready for the instance selection.

3. Proposed Approach

As shown in Figure 1, the proposed approach includes the training and prediction phases. First, the JS divergence values between source and target projects are calculated to select the most similar project. Then, the relative density information is estimated to select the high-density instances. Finally, the selected instances are used to train one-to-one classifiers or many-to-one classifiers. In the prediction phase, the trained classifiers are evaluated by the target project, and the prediction results are obtained. Since the proposed method does not divide the data, cross-validation is not required. The details are illustrated as follows.

3.1. Training Phase

3.1.1. JS divergence for Project Selection. Figure 2 describes the project selection process based on JS divergence. The Gaussian mixture model ($\mathrm{GMM} = \{\mathrm{GMM}_{S1}, \mathrm{GMM}_{S2}, \cdots, \mathrm{GMM}_{Sn}, \mathrm{GMM}_T\}$) is generated first based on the source and target projects. Then, the JS divergence values are acquired based on equations (2) and (3). If the calculated JS divergence is the lowest, it denotes that the source project is the most similar to the target project.

3.1.2. Relative density estimation for instance selection. Since there are still noises and outliers in the source project, high-density instances are required to select. As shown in Figure 3, the number of instances N is counted first. Second, the distance of each instance to its Kth nearest neighbor is calculated to estimate the relative density. Finally, the instances are sorted by percentage and then selected. The parameters of K and percent will be discussed in Section 5.

3.1.3. One-to-One Classifier Training. Some HCPDP models are one-to-one predictions. They use one source project to construct the model and predict the target project. This process is not complex to understand. The learning algorithms can be selected to train the classifier based on the selected instances directly.

3.1.4. Many-to-One Classifiers Training. Besides one-to-one prediction, many-to-one prediction methods aim to add training data by putting all the source projects together. In this paper, a dynamic ensemble voting strategy is proposed to construct classifiers based on JS divergence. As shown in Figure 4, this paper trains the subclassifiers by the selected instances in similar source projects. Then, the percentage of the reciprocal of JS divergence is used as the dynamic voting weight for each subclassifier, that is, $1/JS_i/1/JS_1 + 1/JS_2 + \ldots + 1/JS_{numbers\,of\,source\,projects}$. By using this strategy, the model could guarantee that the selected source projects have the most significant weight, which can strengthen their performance.

3.2. Prediction Phase. For one-to-one prediction, we input the data of the target project into the trained classifier and obtain the prediction label. For many-to-one prediction, the target data is sent to each subclassifier, and then the prediction result is obtained by multiplying the voting strategy. Finally, the result of each classifier is added to get the prediction result. If the prediction result is higher than a threshold of 0.5 (Nam et al. [23]; Zhou et al. [33]; Wan et al. [34]), an instance is more likely to be highly defective.

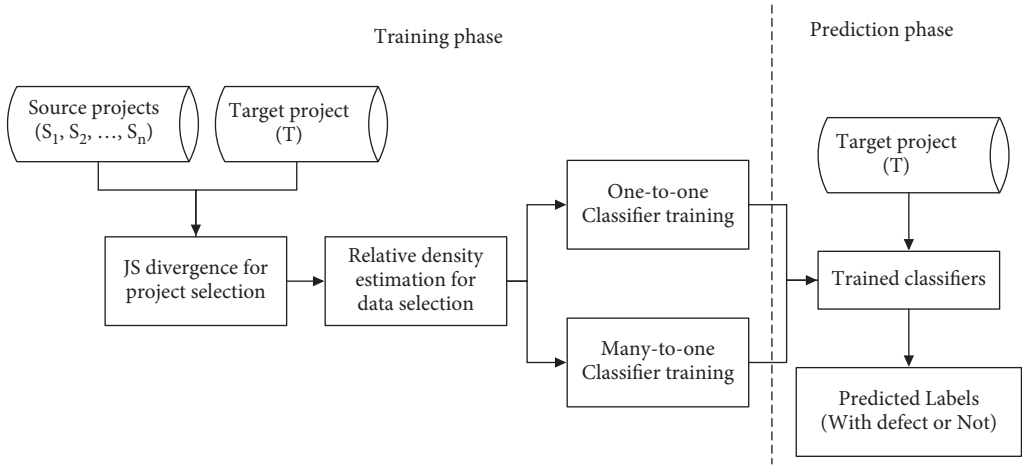

Figure 1: The framework of the proposed approach.

4. Experiment Setup

4.1. Environment and Datasets. The experimental environment is a computer equipped with an Intel Xeon E3-1231 and 16 GB RAM, running Windows 10 (64 bit). Two benchmark datasets, NASA (Shepperd et al. [26]) and PROMISE (Jureczko and Madeyski [27]; Zhou et al. [33]; Xu et al. [35]), are collected. Table 1 shows the total number of instances and defects, and the percentage of defective instances in these datasets.

4.2. Evaluation Metrics. Three wide metrics are used in this study:

4.2.1. F1-Score. Software defect prediction is recognized as a binary classification task. The final result may be True Positive (*TP*), which denotes the number of actually predicted defective modules; False Positive (*FP*) denotes the number of incorrectly predicted non-defective modules; True Negative (*TN*) represents the number of correctly predicted non-defective modules, and False Negative (*FN*) represents the number of incorrectly predicted defective modules. Based on the four results, precision is used to assess the correctness of a prediction model, Precision = $TP/(TP + FP)$, and recall is used to evaluate the possibility of correctly predicted defects, Recall = $TP/(TP + FN)$. In general, there is a trade-off between the two metrics. For example, by sacrificing precision, the recall value (Nam et al. [23]) may be improved. These trade-offs make it difficult to compare the performance of the prediction models using precision and recall (Kim et al. [36]; Xu et al. [35]; Wan et al. [34]). For this reason, this paper compares prediction results using F1-score values, that is, $F1 - \text{score} = 2 \times \text{Precision} \times \text{Recall}/\text{Precision} + \text{Recall}$.

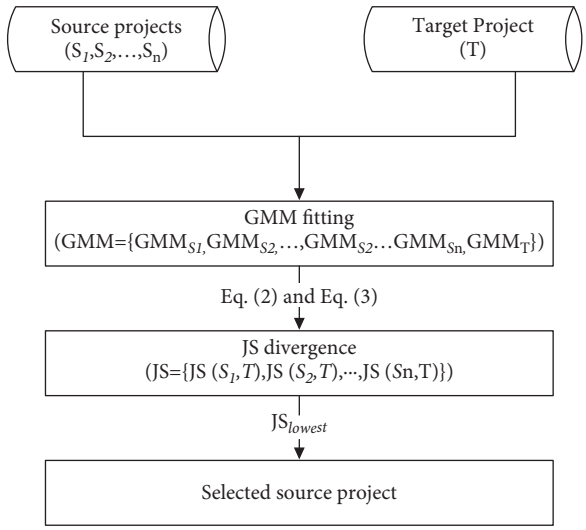

Figure 2: Selection process of the source project.

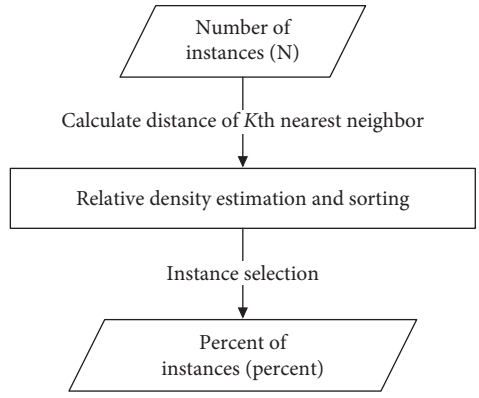

Figure 3: Data selection in source project.

4.2.2. AUC. AUC is used to estimate the area under the receiver operating characteristic curve, obtained by a set of (false positive rate and recall) pairs.

4.2.3. G-Mean. G-mean can reflect the performance while the data is imbalanced, $G - \text{mean} = \sqrt{(TN/TN + FP)(TP/TP + FN)}$.

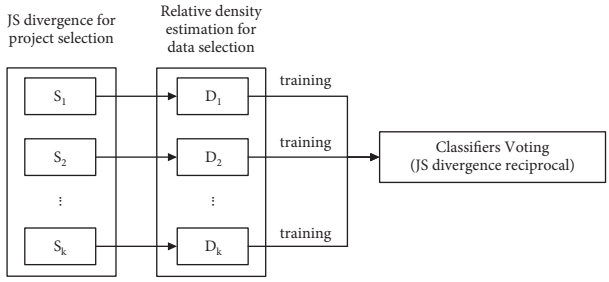

Figure 4: Many-to-one classifiers training.

5. Results and Analysis

In this section, six one-to-one and seven many-to-one HCPDP methods are selected to compare against the proposed approach. Since all the compared methods use logistic regression (LR) (Fan et al. [37]) as the underlying classifier, LR is used to construct the classifier. All the methods were implemented by following their papers.

5.1. Six One-to-One Prediction Methods Are Described as Follows

(i) ManualDown (Zhou et al. [33]). It is a simple unsupervised method, which performs better than most the existing works. ManualDown has been recognized as a new baseline method of CPDP.

(ii) Logistic regression (LR) (Fan et al. [37]). It only constructed an LR classifier to implement one-to-one defect prediction.

(iii) MT (Zhang et al. [38]). The authors proposed Multiple Transformations (MT) to improve prediction performance.

(iv) MT+ (Zhang et al. [38]). The authors improved MT to advance the prediction results.

(v) BDA (Xu et al. [36]). The authors proposed a balanced distribution adaptation (BDA) based transfer learning method to implement cross-project defect prediction.

(vi) CORAL (Sun et al. [38]). A method for domain adaptation was proposed by CORAL, which selected a similar source project with the target project by minimizing the distribution difference.

5.2. Seven Many-to-One Prediction Approaches Are Described as Follows

(i) ManualDown (Zhou et al. [33]). A new baseline method is proposed by calculating the module size.

(ii) Logistic regression (Fan et al. [37]). The LR classifier was constructed to implement many-to-one defect prediction.

(iii) TCA+ (Nam et al. [23]). An algorithm was proposed to deal with the data pre-processing during the many-to-one prediction process.

(iv) TPTL (Liu et al. [24]). A source project estimator was given to select a similar source project and construct the two-phase prediction model.

(v) ISDA (Jing et al. [39]). The idea of subclass discriminant analysis (SDA) was introduced into cross-project defect prediction. The authors improved SDA to advance prediction performance.

(vi) TDS (Herbold et al. [18]). A method for training data selection was given, and the authors used Euclidean distance to measure the difference between the source and target projects.

(vii) DYCOM (Minku et al. [40]). It was a weighted sum of multiple pre-trained models from source projects and 10% of data from the target project.

To check if the proposed approach over other methods was statistically significant, Friedman's test (Friedman [41]) is a favorable statistical test to compare over two ways. At 5% significance level, Friedman's test rejects the null hypothesis of "equal" performance among all the comparing methods. Then Nemenyi's test (Demiar and Schuurmans [42]) is used to analyze which methods differ. Moreover, we also apply Cohen's d (Cohen [43]) to calculate the effect size, which can quantify the difference among the methods. The calculation formula is shown in (5). Table 2 shows the effect grade corresponding to Cohen's d.

$$\text{Cohen}'\text{s } d = \frac{M_1 - M_2}{\sqrt{\left(\sigma_1^2 + \sigma_2^2\right)/2}}. \tag{5}$$

5.3. One-to-One Defect Prediction Results and Analysis. Firstly, Tables 3 and 4 show the JS results of two projects on NASA and PROMISE. Since the JS result is smaller, the two projects are more similar. Then we can select the most similar source project for each target project. In the tables, the smallest JS value is boldface. For example, if CM1 is the target project, PC4 is selected as the source project.

After the source project is determined, instance selection could be completed by following relative density estimation. Then Tables 5–7 present the experimental results (i.e., F1-score, AUC, and G-mean) of the proposed approach compared with the six baseline approaches on NASA and PROMISE. For each target project, the best results are highlighted in bold. The last row in the tables presents the improvement of our approach over other baseline approaches.

Compared with the other six methods on two datasets, the one-to-one prediction method improved F1-score by 3.1%~15.2% in Table 5. In Table 6, AUC is improved by 3.4%~16.4%, and G-mean is improved by 3.5%~20.7% in Table 7. However, the authors ran the approach of MT+ on "poi3.0" many times, but the result was still 0. It may not be suitable to evaluate MT+, and the authors do not also consider this project in their paper.

Next, Friedman's and Nemenyi's tests are used to analyze the performance. First, the corresponding p-values (all less than 0.05) under the Friedman test are given in Table 8,

Table 1: Properties of projects in two data sets.

Dataset	Project	Number of instances	Number of defects	Defect%
NASA	CM1	334	42	12.2
	KC1	2095	325	15.5
	KC3	200	36	18
	MC2	125	44	25.2
	MW1	263	27	10.3
	PC2	1493	16	1.1
	PC4	1379	178	12.9
PROMISE	ant1.7	745	166	22.3
	camel1.6	965	188	19.5
	ivy2.0	352	40	11.4
	jedit4.0	306	75	24.5
	log4j1.0	135	34	25.2
	lucene2.4	340	203	59.7
	poi3.0	442	281	63.6
	synapse1.2	256	86	33.6
	tomcat6.0	858	77	9.0
	velocity1.6	229	78	34.1
	xalan2.4	723	110	15.3
	xerces1.3	453	69	15.3

Table 2: Cohen's d effectiveness level.

Cohen's d	Effectiveness level
$\|d\| < 0.2$	Negligible (N)
$0.2 \le \|d\| < 0.5$	Small (S)
$0.5 \le \|d\| < 0.8$	Medium (M)
$\|d\| \ge 0.8$	Large (L)

Table 3: The calculated JS values of two projects on NASA.

Target	Source						
	CM1	KC1	KC3	MC2	MW1	PC2	PC4
CM1	0	0.6832	0.6795	0.6752	0.6782	0.6864	**0.6418**
KC1	0.6832	0	0.6878	**0.6765**	0.6899	0.6909	0.6883
KC3	0.6795	0.6878	0	**0.6180**	0.6499	0.6842	0.6696
MC2	0.6752	0.6765	**0.6180**	0	0.6839	0.6928	0.6782
MW1	0.6782	0.6899	**0.6499**	0.6839	0	0.6745	0.6835
PC2	0.6864	0.6909	0.6842	0.6928	0.6745	0	**0.6695**
PC4	**0.6418**	0.6883	0.6696	0.6782	0.6835	0.6695	0

which refers to the differences among the seven methods from a global perspective. To further visually show the differences, CD diagrams were used in Figure 5. The average rank of each method is marked along the axis. This study connected them with a thick line if these approaches are not significantly different under the Nemenyi test. By investigating Figure 5 and Table 8, the proposed approach significantly improves the F1-score, AUC, and G-mean over all the other methods.

In Table 9, Cohen's d is used to calculate effect size. The proposed one-to-one defect prediction obtained 14 "L" on NASA datasets, and 9 "L" on PROMISE datasets. It can be concluded that the proposed approach is significantly better than the other six one-to-one prediction methods.

Finally, the parameters of K and percent in Figure 6 are analyzed. To decide the K value, we explore the parameter K based on the number of instances, and the range is $K = \{\lceil \sqrt{N}/4 \rceil, \lceil \sqrt{N}/2 \rceil, \lceil \sqrt{N} \rceil, \lceil 2\sqrt{N} \rceil, \lceil 4\sqrt{N} \rceil\}$, where N is the number of instances. For the percent value, it represents the proportion chosen for sorting instances after relative density estimation, and its range was [0.1, 0.9]. Due to the space limitation, this paper takes CM1 in NASA and ant 1.7 in PROMISE as an example. Then the selected source projects based on JS values are PC4 and xalan2.4, respectively.

Figure 6 plots the variance of the performance with the variance of K and percent. On the X-axis, 0.1~0.9 denotes the proportion of the selected sorting instances. On the Y-axis, 0.25~4 indicates the choice of K, and the Z-axis indicates the values of F1-score, AUC, and G-mean, respectively. As investigated in Figures 6(a)–6(c), when $K = 0.25$, percent = 0.2, the F1-score, AUC, and G-mean of PC4CM1 were 0.3609, 0.6747, and 0.6668. In Figures 6(d)–6(f), when $K = 4$, percent = 0.7, the F1-score, AUC, and G-mean of xalan2.4 ant1.7 were 0.5714, 0.7338, 0.7289. Hence, the selection of K and percent are based on the choice of the project.

5.4. Many-to-One Defect Prediction Results and Analysis. Compared to one-to-one prediction, the proposed many-to-one prediction has the same process for source project selection and instance section. However, the exact number of source projects needed to discuss during the process of the many-to-one prediction, and the corresponding analysis will be given later.

By the ensemble voting weighting strategy Tables 10–12 show the F1-score, AUC, and G-mean of many-to-one prediction under the proposed approach compared with the seven many-to-one methods on NASA and PROMISE. The best results are still highlighted in bold, and the last row presents the improvement of the proposed approach over other approaches.

According to the results in Table 10, the proposed many-to-one prediction improved the F1-score by 6.7%~22.6%. In

Table 4: The calculated JS values of two projects on PROMISE.

Target	Source											
	ant1.7	camel1.6	ivy2.0	jedit4.0	log4j1.0	lucene2.4	poi3.0	synapse1.2	Tomcat	velocity1.6	xalan2.4	xerces1.3
ant1.7	0	0.6699	0.6802	0.6929	0.6892	0.6839	0.6810	0.6871	0.6929	0.6890	**0.6616**	0.6858
camel1.6	0.6699	0	0.6873	0.6928	0.6801	0.6731	0.6850	0.6632	0.6926	0.6820	**0.6363**	0.6908
ivy2.0	0.6802	0.6873	0	0.6924	0.6841	**0.6690**	0.6919	0.6861	0.6930	0.6849	0.6835	0.6871
jedit4.0	0.6929	0.6928	0.6924	0	0.6931	0.6916	0.6930	0.6931	0.6931	0.6930	0.6927	**0.6911**
log4j1.0	0.6892	0.6801	0.6841	0.6931	0	**0.6750**	0.6915	0.6753	0.6929	0.6767	0.6775	0.6916
lucene2.4	0.6839	0.6731	0.6690	0.6916	0.6750	0	0.6877	**0.6569**	0.6927	0.6873	0.6750	0.6898
poi3.0	**0.6810**	0.6850	0.6919	0.6930	0.6915	0.6877	0	0.6873	0.6929	0.6921	0.6905	0.6913
synapse1.2	0.6871	0.6632	0.6861	0.6931	0.6753	**0.6569**	0.6873	0	0.6927	0.6918	0.6702	0.6919
Tomcat	0.6929	0.6926	0.6930	0.6931	0.6929	0.6927	0.6929	0.6927	0	0.6931	**0.6923**	0.6931
velocity1.6	0.6890	0.6820	0.6849	0.6930	**0.6767**	0.6873	0.6921	0.6918	0.6931	0	0.6826	0.6890
xalan2.4	0.6616	**0.6363**	0.6835	0.6927	0.6775	0.6750	0.6905	0.6702	0.6923	0.6826	0	0.6880
xerces1.3	**0.6858**	0.6908	0.6871	0.6911	0.6916	0.6898	0.6913	0.6919	0.6931	0.6890	0.6880	0

Table 5: F1-score of the proposed approach and six one-to-one prediction methods on NASA and PROMISE.

Target		Ours	ManualDown	LR	MT	MT+	BDA	CORAL
NASA	CM1	**0.3609**	0.3062	0.2551	0.1935	0.1880	0.2932	0.2549
	KC1	**0.4451**	0.4127	0.2847	0.1916	0.2831	0.4031	0.3526
	KC3	**0.3830**	0.3485	0.3254	0.2273	0.2545	0.3807	0.3810
	MC2	**0.5882**	0.5283	0.3867	0.2525	0.2857	0.4692	0.4214
	MW1	**0.3443**	0.2658	0.1833	0.1399	0.1524	0.3201	0.2446
	PC2	**0.0732**	0.0306	0.0550	0.0295	0.0322	0.0770	0.0571
	PC4	**0.3286**	0.3011	0.3069	0.1133	0.2554	0.2967	0.3015
PROMISE	ant1.7	**0.5714**	0.5326	0.5102	0.3443	0.3831	0.5246	0.5100
	camel1.6	**0.3849**	0.3463	0.3292	0.2889	0.2940	0.3449	0.3273
	ivy2.0	0.3409	0.3271	0.3513	0.2010	0.2132	**0.3666**	0.3539
	jedit4.0	**0.5745**	0.4978	0.4897	0.3690	0.3667	0.5107	0.4931
	log4j1.0	0.5366	0.5347	0.5283	0.2936	0.3248	**0.5956**	0.5425
	lucene2.4	0.6506	0.6237	0.5225	0.6705	**0.7421**	0.5726	0.5437
	poi3.0	0.6427	0.7094	0.5610	0.6943	**0.7773**	0.5340	0.5774
	synapse1.2	**0.5946**	0.5943	0.5645	0.4566	0.4751	0.5782	0.5696
	Tomcat	**0.4179**	0.2653	0.2938	0.1612	0.1456	0.3268	0.3058
	velocity1.6	0.5000	**0.5521**	0.4372	0.4223	0.4848	0.4605	0.4516
	xalan2.4	**0.4345**	0.4119	0.3625	0.2816	0.2717	0.3948	0.3669
	xerces1.3	0.4000	0.3878	0.3924	0.3545	**0.4252**	0.3568	0.3851
Average		**0.4512**	0.4198	0.3758	0.2992	0.3345	0.4108	0.3916
Improvement			**3.1%**	**7.5%**	**15.2%**	**11.7%**	**4.0%**	**6.0%**

Table 11, AUC is improved by 2.3%~7.7%, and G-mean is improved by 5.8%~22.8% in Table 12 on NASA and PROMISE. In these tables, some results are still 0, the relevant project may not be suitable to evaluate, and the authors do not consider the projects in their papers.

Table 13 gives the p-values (all less than 0.05), which indicate that all the methods have significant performance differences between each other. Moreover, the CD diagrams in Figure 7 show that the proposed approach ranks first in all the evaluation metrics, except ranking second in G-mean on NASA and AUC on PROMISE. As a whole, many-to-prediction under the proposed approach significantly improved the F1-score, AUC, and G-mean over other many-to-one methods.

Similarly, Cohen's d is used to evaluate effect size. The analysis results can be seen in Table 14 and the proposed many-to-one defect prediction obtained 10 "L" on NASA datasets, and 16 "L" on PROMISE datasets. The proposed approach is significantly better than the other seven many-to-one prediction methods, and the difference cannot be ignored.

Finally, Figure 8 illustrates the variance of the evaluation metrics with the numbers of the source projects. It can be found that when the number of NASA source projects is 3 and the number of PROMISE source projects are 7, the result is optimal. For example, if "CM1" is the target project, the selected source projects are PC4, MC2, and MW1.

5.5. Different Learning Methods Analysis. This section aims to prove that the proposed is not subject to machine learning methods, that is, the prediction performance can be improved regardless of which learning method is selected. We first select four traditional learning methods that are often applied in CPDP, and they are LR, SVM, KNN, RF, and NB, respectively.

Since AUC can reveal the classification performance, we take it as an example. Figure 9 shows the comparison results, and it can be seen that AUC is improved regardless of which type of learning method is used to construct the classifier.

Table 6: AUC of the proposed methods and six one-to-one prediction approaches on NASA and PROMISE.

Target		Ours	ManualDown	LR	MT	MT+	BDA	CORAL
NASA	CM1	**0.6748**	0.6574	0.5827	0.5193	0.5250	0.6100	0.5824
	KC1	**0.7225**	0.7183	0.5547	0.5132	0.5403	0.6562	0.6254
	KC3	**0.6280**	0.5969	0.5723	0.5183	0.5115	0.6243	0.6202
	MC2	**0.6853**	0.6083	0.5412	0.5271	0.5621	0.6130	0.5665
	MW1	**0.7321**	0.6558	0.5237	0.5438	0.5575	0.6804	0.6075
	PC2	**0.8404**	0.6098	0.7592	0.5690	0.5793	0.7911	0.7698
	PC4	**0.6460**	0.6334	0.6148	0.5588	0.5723	0.6049	0.6095
PROMISE	ant1.7	**0.7338**	0.7338	0.6961	0.5188	0.5946	0.7063	0.6959
	camel1.6	**0.6168**	0.5730	0.5663	0.5158	0.5209	0.5806	0.5654
	ivy2.0	0.7051	**0.7147**	0.6970	0.5143	0.5372	0.7138	0.7005
	jedit4.0	**0.7323**	0.6699	0.6579	0.5228	0.5013	0.6769	0.6612
	log4j1.0	0.6948	0.6990	0.6879	0.5209	0.5374	**0.7391**	0.7000
	lucene2.4	**0.6894**	0.5923	0.5824	0.5068	0.5037	0.6351	0.5995
	poi3.0	**0.6892**	0.6876	0.6136	0.5174	0.5000	0.6265	0.6335
	synapse1.2	**0.6904**	0.6810	0.6646	0.5207	0.5281	0.6793	0.6705
	Tomcat	**0.7772**	0.7040	0.6947	0.5043	0.5320	0.7308	0.7117
	velocity1.6	0.6209	**0.6378**	0.5715	0.5381	0.6008	0.5957	0.5812
	xalan2.4	0.6990	**0.7256**	0.6433	0.5551	0.5416	0.6738	0.6477
	xerces1.3	0.6632	**0.6943**	0.6680	0.6342	0.7207	0.6393	0.6607
Average		**0.6969**	0.6628	0.6259	0.5326	0.5509	0.6620	0.6426
Improvement			**3.4%**	**7.1%**	**16.4%**	**14.6%**	**3.5%**	**5.4%**

Table 7: G-mean of the proposed approach and six one-to-one prediction methods on NASA and PROMISE.

Target		Ours	ManualDown	LR	MT	MT+	BDA	CORAL
NASA	CM1	**0.6668**	0.6491	0.5612	0.4651	0.4543	0.5814	0.5627
	KC1	**0.7225**	0.7039	0.5114	0.3332	0.3961	0.6270	0.5947
	KC3	**0.6149**	0.5954	0.5605	0.4555	0.4966	0.6050	0.6083
	MC2	**0.6752**	0.6077	0.5078	0.3784	0.4140	0.5779	0.5382
	MW1	**0.7307**	0.6444	0.5062	0.4215	0.4162	0.6777	0.5966
	PC2	**0.8348**	0.6049	0.7552	0.5366	0.5617	0.7902	0.7654
	PC4	**0.6430**	0.6268	0.6017	0.3129	0.5021	0.5759	0.5942
PROMISE	ant1.7	**0.7338**	0.7338	0.6961	0.5188	0.5946	0.7063	0.6959
	camel1.6	**0.6168**	0.5730	0.5663	0.5158	0.5209	0.5806	0.5654
	ivy2.0	0.7051	**0.7147**	0.6970	0.5143	0.5372	0.7138	0.7005
	jedit4.0	**0.7323**	0.6699	0.6579	0.5228	0.5013	0.6769	0.6612
	log4j1.0	0.6948	0.6990	0.6879	0.5209	0.5374	**0.7391**	0.7000
	lucene2.4	**0.6894**	0.5923	0.5824	0.5068	0.5037	0.6351	0.5995
	poi3.0	**0.6892**	0.6876	0.6136	0.5174	0.5000	0.6265	0.6335
	synapse1.2	**0.6904**	0.6810	0.6646	0.5207	0.5281	0.6793	0.6705
	Tomcat	**0.7772**	0.7040	0.6947	0.5043	0.5320	0.7308	0.7117
	velocity1.6	0.6209	**0.6378**	0.5715	0.5381	0.6008	0.5957	0.5812
	xalan2.4	0.6990	**0.7256**	0.6433	0.5551	0.5416	0.6738	0.6477
	xerces1.3	0.6632	**0.6943**	0.6680	0.6342	0.7207	0.6393	0.6607
Average		**0.6947**	0.6603	0.6183	0.4880	0.5189	0.6543	0.6362
Improvement			**3.5%**	**7.7%**	**20.7%**	**17.6%**	**4.0%**	**5.9%**

Table 8: The p-values under Friedman's test among the seven methods.

Data set	p-value$_{\text{F1-score}}$	p-value$_{\text{AUC}}$	p-value$_{\text{G-mean}}$
NASA	7.99E-06	4.64E-06	2.10E-06
PROMISE	1.30E-04	6.12E-09	2.32E-08

To check the difference in learning methods, Figure 10 graphically visualizes the average comparison results of the proposed approach with five learning methods on two datasets. When the classifier changes, the proposed approach still works well. It proves that the proposed approach has more robust adaptability compared to other learning methods. However, we cannot easily judge which learning method is better because it is affected by the datasets.

5.6. Discussion about the Time-Efficiency. To illustrate whether the running time is acceptable, Tables 15 and 16 give the average running time of our approach and other approaches. Compared with TPTL and ISDA, the proposed approach needs less running time, but more than the other methods.

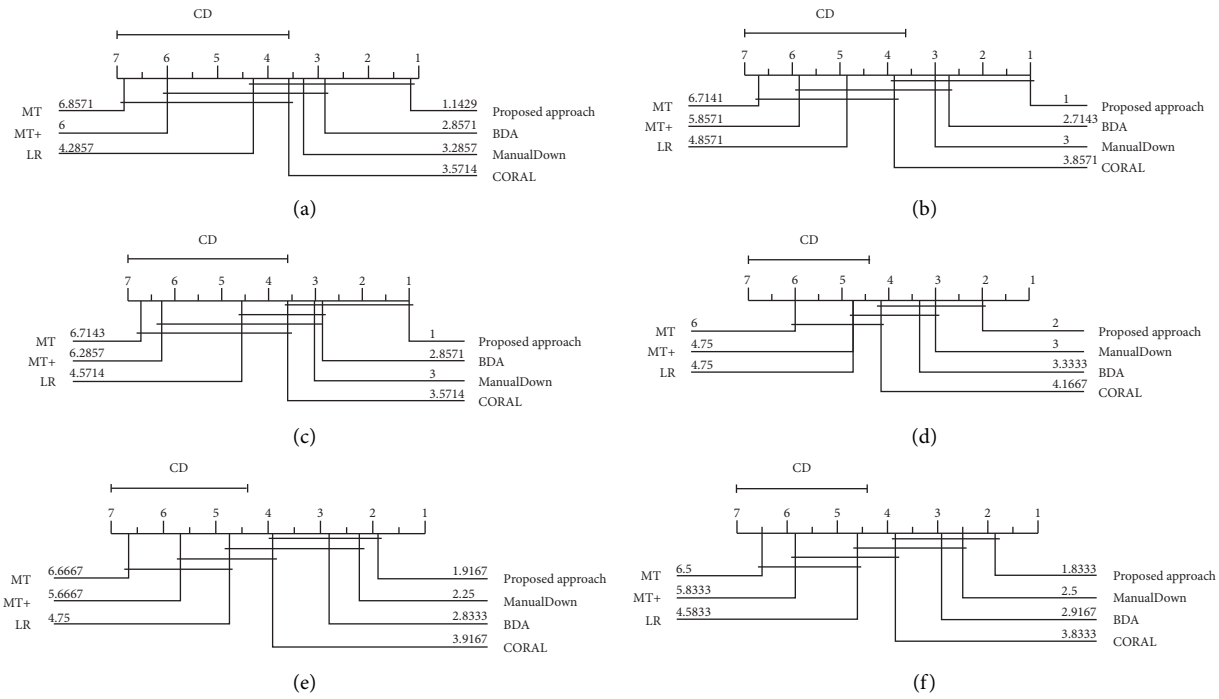

Figure 5: One-to-one comparison of average ranks in F1-score, AUC, and G-mean on two datasets. (a) NASA: F1-score. (b) NASA: AUC. (c) NASA: G-mean. (d) PROMISE: F1-score. (e) PROMISE: AUC. (f) PROMISE: G-mean.

Table 9: Cohen's d of our approach and six one-to-one prediction approaches on NASA and PROMISE.

Dataset	Metrics	Manualdown	LR	MT	MT+	BDA	CORAL
NASA	F1-score	0.3316 (S)	0.8382 (L)	1.7424 (L)	1.3012 (L)	0.3113 (S)	0.5680 (M)
	AUC	1.1918 (L)	1.6040 (L)	3.4742 (L)	3.1416 (L)	0.7857 (M)	1.2284 (L)
	G-mean	1.2166 (L)	1.6853 (L)	4.0312 (L)	3.8117 (L)	0.9263 (L)	1.3265 (L)
PROMISE	F1-score	0.1913 (N)	0.6076 (M)	0.9499 (L)	0.6422 (M)	0.4063 (S)	0.5259 (M)
	AUC	0.3613 (S)	1.0423 (L)	4.1741 (L)	2.7089 (L)	0.5657 (M)	0.8960 (L)
	G-mean	0.3795 (S)	0.9651 (L)	2.6940 (L)	1.7510 (L)	0.5961 (M)	0.8471 (L)

It is necessary to explain why our approach needs more time. Four components are contained in our approach, which includes JS calculation for source project selection, relative density estimation for instance selection, classifier building, and prediction. The running time of classifier building and prediction is less, and they can be ignored. In Table 17, it can be seen that more time is used for JS calculation and the relative density estimation. As relative density estimation is based on the selected project, JS calculation is the main reason our approach needs more time. However, the calculation process does not need to be updated all the time. Even if a new project is added to the datasets, we only need to calculate the JS values of the new project with other projects, instead of recalculating all the JS values.

6. Threats to Validity

Four potential threats to the validity are described in the following:

(1) Accuracy of experiments. Most of the compared works do not provide codes of their methods. This study only analyses and implements their methods by following their papers.

(2) Bias of evaluation measures. In this work, the wide measures of F1-score, AUC, and G-mean are used to show the results of the prediction. Other measures, such as recall, precision, skewed F-measure, and Matthews correlation coefficient, are not considered.

(3) Bias of classifiers. Classification is a significant research topic, and many learning methods can be used to build classifiers. As investigated in previous studies, Logistic regression can achieve better performance. Therefore, this work also applied logistic regression to build the classifiers. Meanwhile, the convention learning methods of SVM, KNN, Random Forest, and Naïve Bayes are also tested to evaluate the performance.

(4) Bias of datasets. Several benchmark data sets are commonly used in the field of cross-project defect prediction, such as NASA (Shepperd et al. [26]), PROMISE (Jureczko and Madeyski [27]), AEEEM (D'Ambros et al. [44]), Relink (Wu et al. [45]), and

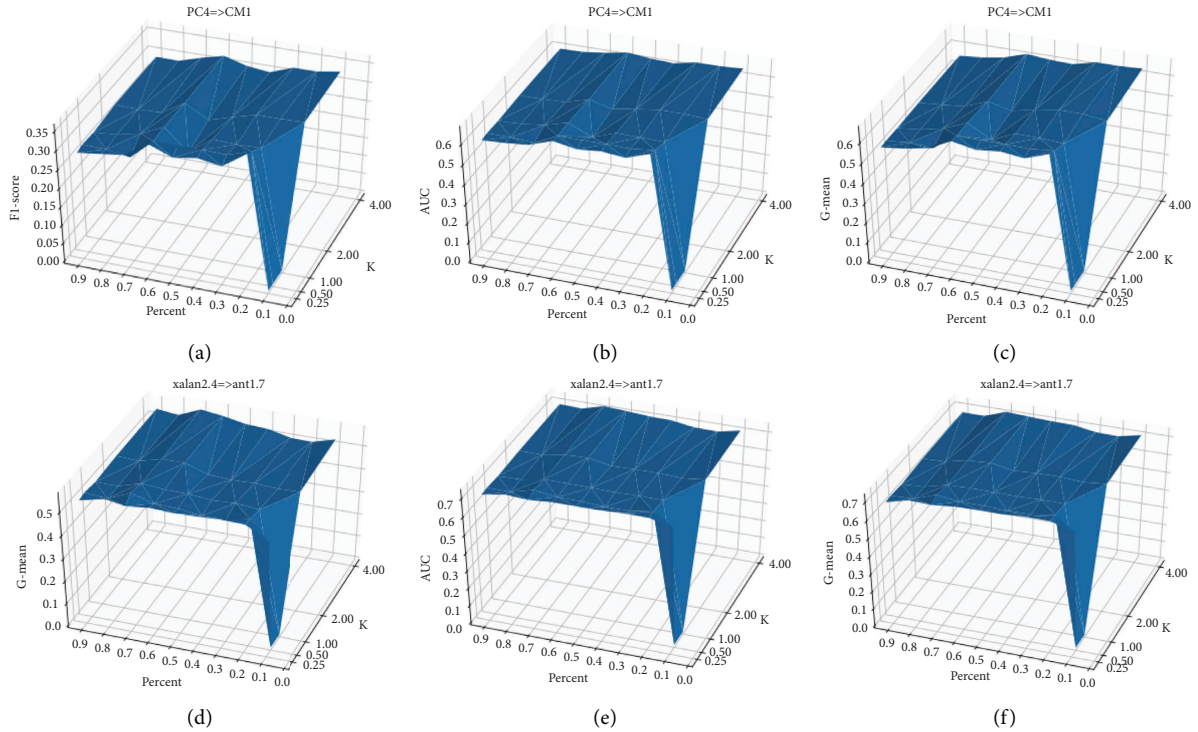

Figure 6: The variance of the proposed approach performance with (K) and percent. (a) F1-score of PC4⇒CM1. (b) AUC of PC4⇒CM1. (c) G-mean of PC4⇒CM1. (d) F1-score of xalan2.4⇒ant1.7. (e) AUC of xalan2.4⇒ant1.7. (f) G-mean of xalan2.4⇒ant1.7.

Table 10: F1-score of the proposed approach and seven many-to-one prediction methods on two datasets.

Target		Ours	ManualDown	LR	MT	MT+	BDA	CORAL
NASA	CM1	0.3357	0.3265	0.2636	0.3198	0.3121	**0.3713**	0.2222
	KC1	**0.4471**	0.2723	0.2154	0.3935	0.3174	0.3435	0.1991
	KC3	**0.4941**	0.3207	0.3958	0.3259	0.3932	0.4423	0.2449
	MC2	0.5814	0.3095	0.3000	0.4401	0.5688	**0.6000**	0.0000
	MW1	**0.3415**	0.3308	0.2185	0.2992	0.2821	0.2899	0.0588
	PC2	0.0785	**0.3483**	0.0606	0.0628	0.0132	0.0782	0.0625
	PC4	**0.3535**	0.3430	0.3440	0.2097	0.3030	0.2596	0.1494
PROMISE	ant1.7	**0.5742**	0.4830	0.5673	0.5408	0.5070	0.4990	0.4990
	camel1.6	0.3774	**0.4994**	0.3103	0.3294	0.3862	0.3859	0.1017
	ivy2.0	0.4324	**0.4897**	0.3916	0.3448	0.3125	0.3314	0.4062
	jedit4.0	**0.6215**	0.4832	0.5150	0.4905	0.5185	0.4257	0.4567
	log4j1.0	**0.6750**	0.4826	0.5952	0.5167	0.5417	0.5275	0.3182
	lucene2.4	**0.6923**	0.4640	0.4730	0.5257	0.5714	0.6138	0.6322
	poi3.0	0.7474	0.4390	0.5902	0.5246	0.7713	**0.8066**	0.2310
	synapse1.2	**0.6703**	0.4781	0.5810	0.6069	0.6071	0.4550	0.3793
	Tomcat	0.3718	**0.5129**	0.3385	0.3095	0.2590	0.2145	0.2136
	velocity1.6	0.5548	0.4799	0.5098	0.4654	0.5083	**0.6145**	0.1176
	xalan2.4	0.4469	**0.4963**	0.4085	0.4036	0.3573	0.2874	0.3290
	xerces1.3	**0.5294**	0.4879	0.4082	0.4296	0.3945	0.4636	0.4107
Average		**0.4908**	0.4235	0.3940	0.3968	0.4171	0.4216	0.2648
Improvement			**6.7%**	**9.7%**	**9.4%**	**7.4%**	**6.9%**	**22.6%**

SOFTLAB (Turhan et al. [17]), and some datasets contain different versions of each project. To evaluate the prediction performance, this work chose the widely used NASA and PROMISE as the datasets.

7. Related Work

Cross-project defect prediction has become one of the most important topics in software engineering (Rahman et al. [2]; Yang et al. [3]; Ghotra et al. [4]; Herbold et al. [14]; Wen et al.

Table 11: AUC of the proposed approach and seven many-to-one prediction methods on two datasets.

Target		Ours	ManualDown	LR	MT	MT+	BDA	CORAL
NASA	CM1	0.6582	0.7165	0.5865	**0.7168**	0.6954	0.7134	0.5131
	KC1	0.7155	0.7097	0.5381	0.7360	0.6756	0.6316	0.5746
	KC3	0.7063	**0.7089**	0.6389	0.5992	0.6306	0.6822	0.5183
	MC2	0.6792	**0.7031**	0.5118	0.6130	0.6748	0.6987	0.6622
	MW1	0.7300	0.7132	0.5734	0.7521	**0.7349**	0.6776	0.5403
	PC2	0.8499	0.6523	0.8117	**0.8966**	0.7043	0.8724	0.6648
	PC4	0.6655	**0.7469**	0.6522	0.5205	0.6511	0.5764	0.5316
PROMISE	ant1.7	0.7457	0.6549	0.7414	**0.7864**	0.6556	0.6952	0.7574
	camel1.6	0.6106	**0.6640**	0.5526	0.5842	0.6312	0.6171	0.5561
	ivy2.0	0.7782	0.6556	0.7298	0.7065	0.6794	0.6881	0.7683
	jedit4.0	**0.7649**	0.6585	0.6806	0.7050	0.6656	0.5914	0.6885
	log4j1.0	**0.8030**	0.6580	0.7439	0.7616	0.6622	0.6896	0.7312
	lucene2.4	**0.7225**	0.6657	0.5885	0.6494	0.5389	0.5704	0.6772
	poi3.0	0.7493	0.6532	0.6621	0.7349	0.6990	**0.7504**	0.7432
	synapse1.2	0.7516	0.6582	0.6817	**0.7725**	0.7476	0.5735	0.7201
	Tomcat	**0.7633**	0.6596	0.7336	0.7553	0.7466	0.5993	0.7232
	velocity1.6	0.6631	0.6597	0.6308	0.6411	0.5033	**0.7002**	0.6825
	xalan2.4	0.7266	0.6573	0.6814	**0.7519**	0.6655	0.5674	0.7078
	xerces1.3	**0.7858**	0.6545	0.6766	0.7553	0.7824	0.7394	0.7834
Average		**0.7300**	0.6763	0.6535	0.7073	0.6707	0.6650	0.6602
Improvement			**5.4%**	**7.7%**	**2.3%**	**5.9%**	**6.5%**	**7.0%**

Table 12: G-mean of the proposed approach and seven many-to-one prediction methods on two datasets.

Target		Ours	ManualDown	LR	MT	MT+	BDA	CORAL
NASA	CM1	0.6525	0.6984	0.5576	0.4494	0.4377	**0.7130**	0.4876
	KC1	**0.7148**	0.6930	0.4235	0.5104	0.4325	0.6284	0.6096
	KC3	0.6955	0.6958	0.6292	0.4628	0.5029	0.6809	0.6225
	MC2	0.6700	0.6891	0.4381	0.5393	0.6112	0.6816	0.0000
	MW1	**0.7284**	0.7008	0.5660	0.4225	0.4052	0.6746	0.3583
	PC2	0.8454	0.6461	0.8019	0.1796	0.0815	**0.8630**	0.2487
	PC4	0.6612	**0.7306**	0.6433	0.3516	0.4250	0.5624	0.5010
PROMISE	ant1.7	**0.7449**	0.6502	0.7408	0.6258	0.5864	0.6927	0.5833
	camel1.6	0.6032	**0.6591**	0.5390	0.4571	0.4859	0.6167	0.4495
	ivy2.0	**0.7779**	0.6514	0.7292	0.5318	0.4330	0.6880	0.7050
	jedit4.0	**0.7643**	0.6537	0.6721	0.6027	0.5928	0.5801	0.6758
	log4j1.0	**0.8030**	0.6533	0.7438	0.6036	0.6111	0.6894	0.7408
	lucene2.4	**0.7076**	0.6594	0.5357	0.6000	0.5766	0.5704	0.6299
	poi3.0	0.7424	0.6472	0.6267	0.6137	0.7151	**0.7495**	0.5508
	synapse1.2	**0.7510**	0.6531	0.6774	0.6797	0.6462	0.5688	0.7250
	Tomcat	**0.7632**	0.6557	0.7333	0.4323	0.3856	0.5979	0.6241
	velocity1.6	0.6536	0.6545	0.6171	0.5681	0.5772	**0.7002**	0.6924
	xalan2.4	**0.7266**	0.6531	0.6775	0.5172	0.4662	0.5620	0.4429
	xerces1.3	**0.7858**	0.6502	0.6696	0.5476	0.4957	0.7394	0.5233
Average		**0.7259**	0.6681	0.6327	0.5103	0.4983	0.6610	0.5353
Improvement			**5.8%**	**9.3%**	**21.6%**	**22.8%**	**6.5%**	**19.1%**

Table 13: The p-values under Friedman test of our approach and seven many-to-one prediction approaches.

Data set	p-value$_{\text{F1-score}}$	p-value$_{\text{AUC}}$	p-value$_{\text{G-mean}}$
NASA	2.38E-04	3.31E-02	8.94E-05
PROMISE	1.30E-04	5.21E-05	2.56E-06

[5]). In practice, researchers have recognized it as a binary classification problem, that is, they model the defective data to train a machine learning model. Due to insufficient training data in the source projects, the researchers try to build a bridge from the source project to the target project. They utilize the information from the source projects and transfer it to the target project. The following will give the related studies only.

For one-to-one defect prediction, Sun et al. [38] described an effective and efficient method for domain adaptation called CORAL, which minimized the distribution difference between source and target projects. Zhang et al. [46] applied different transformations to explore whether the cross-project defect prediction is affected and proposed Multiple Transformations (MT) and MT+ to improve

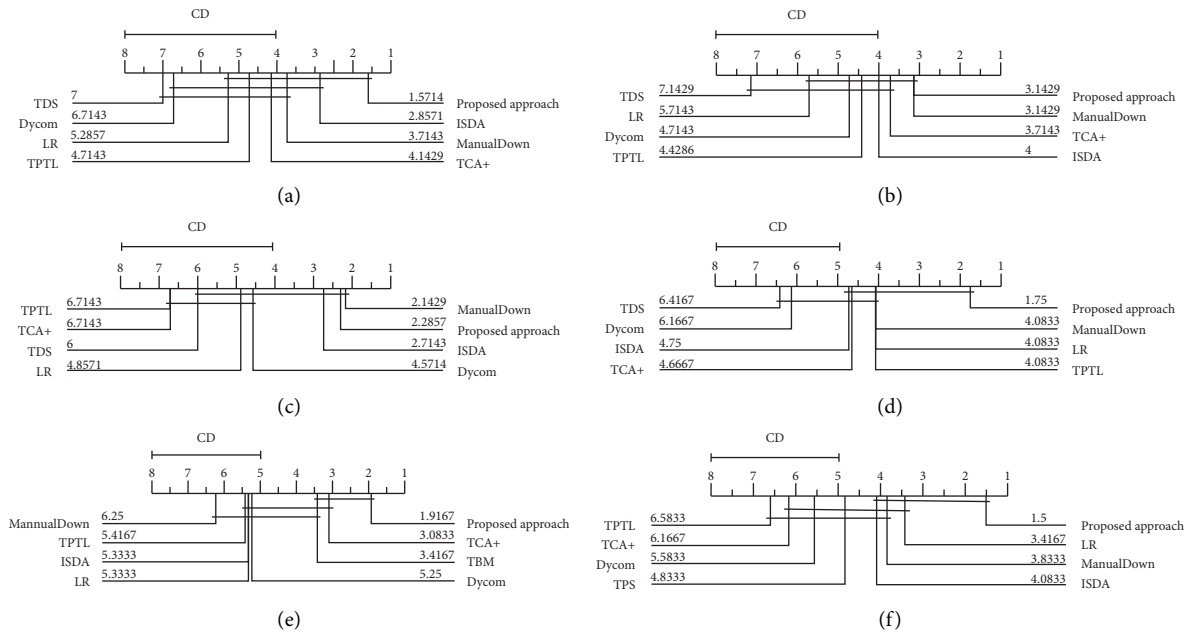

Figure 7: Many-to-one comparison of average ranks in F1-score, AUC, and G-mean on two datasets. (a) NASA: F1-score. (b) NASA: AUC. (c) NASA: G-mean. (d) PROMISE: F1-score. (e) PROMISE: AUC. (f) PROMISE: G-mean.

Table 14: Cohen's d of our approach and seven many-to-one prediction approaches on NASA dataset.

Dataset	Metrics	ManualDown	LR	MT	MT+	BDA	CORAL
NASA	F1-score	0.5137 (M)	0.9428 (L)	0.6253 (M)	0.4204 (S)	0.2374 (S)	1.9947 (L)
	AUC	0.1657 (N)	1.2675 (L)	0.2657 (S)	0.7041 (M)	0.2958 (S)	2.3613 (L)
	G-mean	0.3527 (S)	1.3615 (L)	3.2539 (L)	2.5772 (L)	0.3157 (S)	2.0158 (L)
PROMISE	F1-score	0.8531 (L)	0.7585 (M)	0.9331 (L)	0.6151 (M)	0.6386 (M)	1.5802 (L)
	AUC	2.1955 (L)	1.1560 (L)	0.3895 (S)	1.1235 (L)	1.5271 (L)	0.4996 (S)
	G-mean	2.1127 (L)	1.1506 (L)	2.7476 (L)	2.4973 (L)	1.4508 (L)	1.5486 (L)

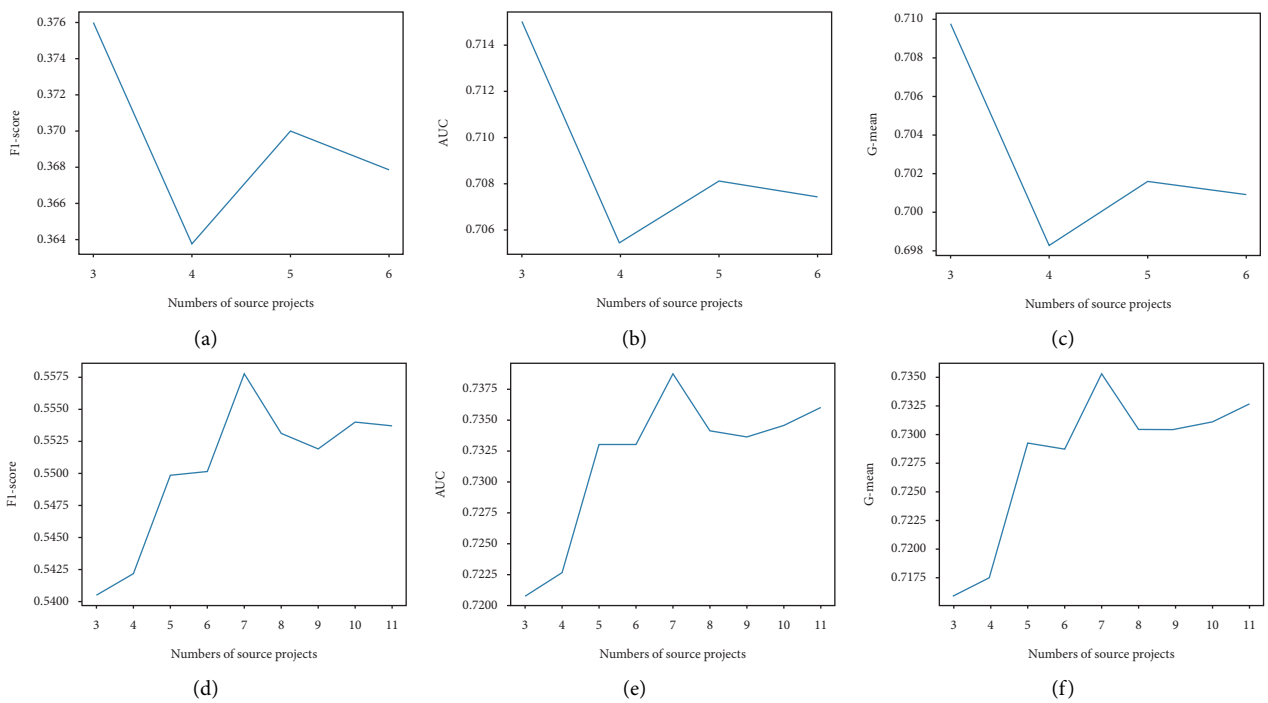

Figure 8: The variance of the evaluation metrics with the numbers of the source projects. (a) F1-score of NASA. (b) AUC of NASA. (c) G-mean of NASA. (d) F1-score of PROMISE. (e) AUC of PROMISE. (f) G-mean of PROMISE.

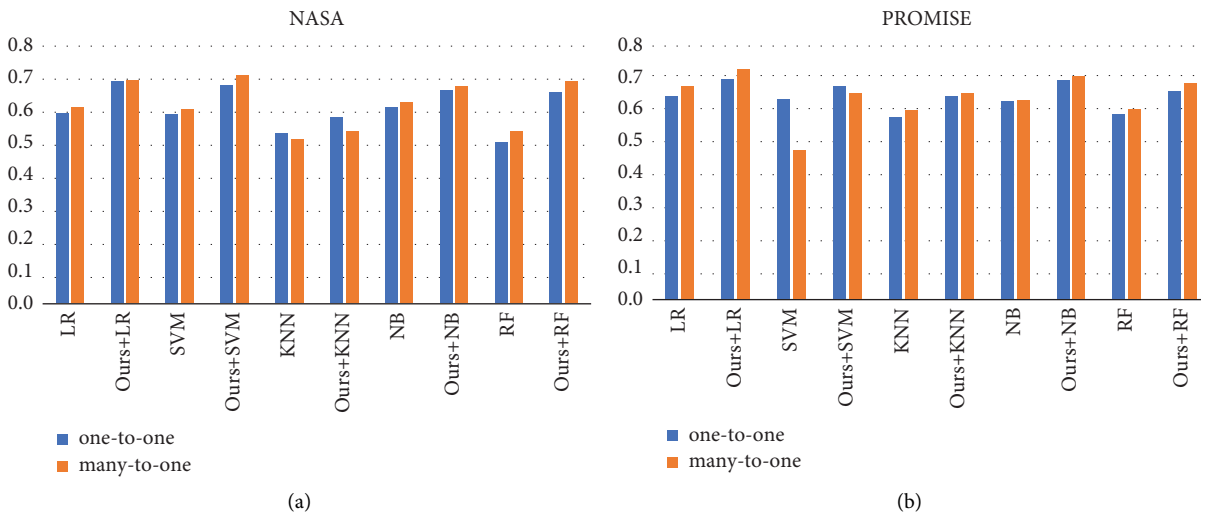

Figure 9: Learning methods comparison results in AUC on two datasets. (a) NASA. (b) PROMISE.

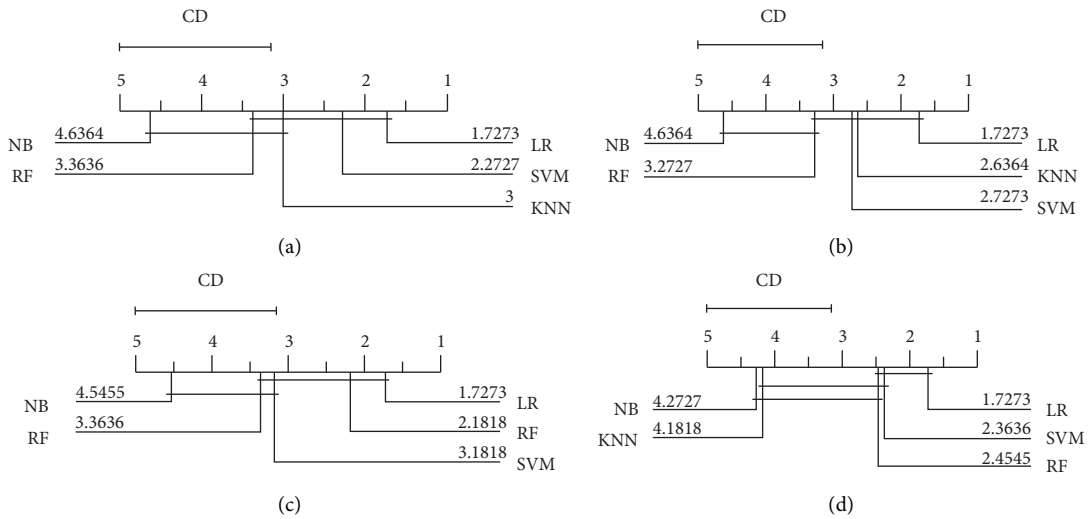

Figure 10: Comparison of average ranks in F1-score on NASA and PROMISE. (a) One-to-one prediction on NASA. (b) Many-to-one prediction on NASA. (c) One-to-one prediction on PROMISE. (d) Many-to-one prediction on PROMISE.

Table 15: Average running time of one-to-one approaches on NASA and PROMISE.

Data set	Running time of approach (*second*)					
	Ours	LR	MT	MT+	BDA	Coral
NASA	228.6	4.4	29.5	36.2	70.5	5.0
PROMISE	528.2	7.7	58.2	79.2	104.8	9.6

Table 16: Average running time of many-to-one approaches on NASA and PROMISE.

Data set	Running time of approach (*second*)							
	Ours	LR	TCA+	TPTL	ISDA	Lt	TDS	Dycom
NASA	524.2	2.7	8.1	2420.6	12495.7	0.2	0.2	4.1
PROMISE	924.0	1.7	14.1	1395.2	18830.4	0.4	0.3	5.9

Table 17: Average running time of JS calculation and relative density estimation on NASA and PROMISE.

Component	One-to-one		Many-to-one	
	NASA	PROMISE	NASA	PROMISE
JS calculation	163.7	483.2	163.7	483.2
Relative density estimation	64.1	36.2	359.8	440.3
Total	227.8	519.4	523.5	923.5

prediction performance. Xu et al. [35] introduced a balanced distribution adaptation-based transfer learning method to implement defect prediction, which improved the existing methods' performance. In this study, we compare those one-to-one models with our one-to-one approach.

For many-to-one defect prediction, Herbold [18] utilized distance-based strategies to select training data in many-to-one defect prediction. The results show their method can achieve good performance. Nam et al. [23] extended their work to TCA+ with a preprocessing data method and conducted it on one-to-one and many-to-one predictions. Minku et al. [40] proposed a transfer learning model for Dycom based on the work (Nam et al. [23]). Dycom is a weighted sum of transferred models trained by various source projects. Jing et al. [39] proposed a method called subclass discriminant analysis, which can learn features from original metrics and make the distributions of source and target projects stable. Liu et al. [24] proposed a two-phase transfer learning model (TPTL) for CPDP. They introduced a source project estimator to choose similar source projects and built the two prediction models. By combining the prediction results of the two models, they further improved the final performance. In this study, we compare the above models with our many-to-one approach.

Besides homogeneous defect prediction, heterogeneous defect prediction has recently become a great process. Gong et al. [47] utilized the thought of stratification embedded in the nearest neighbor to produce evolving training datasets with balanced data. Zou et al. [48] proposed a method named Joint Feature representation with double marginalized denoising auto-encoders to learn the global and local features, and they introduced local data gravitation between source and target domains to determine instance weight in the learning process (Zou et al. [49]). Jin et al. [50] used two support vector machines to implement domain adaptation to match data distribution. Mehta and Patnaik [51] used various ensemble machine learning techniques to improve classification performance. In the future, we will study heterogeneous defect prediction based on the state-of-the-art model.

From the abovementioned literature, it can be concluded that there are two significant problems in CPDP research. The first problem is that the source and target project data usually exhibit significantly different distributions. Since the source and target projects might be implemented by different programming languages or companies, the same metrics in the source and target project data might have different distributions. The second problem is that there are no standard metrics between the source and target project data. It is difficult to predict the defects in the target project using conventional methods to build models on the source project data. Therefore, many efforts have been made to solve these two problems in recent years, and this study focuses on the solution to the first problem. In the future, standardized tools will be developed based on more experiments.

8. Conclusion and Future Works

This paper studies the limitations of prior studies and proposes a novel approach. First, to avoid the randomness of the source project selection, Jensen-Shannon divergence is used to measure the similarity between source and target projects automatically. Subsequently, relative density information is introduced to filter noise and outliers in similar source projects. Posteriorly, the one-to-one defect prediction model is constructed based on the most similar source project directly, and the many-to-one defect prediction model is built by a proposed ensemble weight strategy based on Jensen-Shannon divergence. Finally, the models predict the potential defects in the target project.

Experimental results on the benchmark datasets of NASA and PROMISE indicate that our approach can improve the F1-score, AUC, and G-mean. Moreover, the results also prove that the proposed approach is adaptive regardless of the type of learning method.

In further work, we will select more defect data sets to evaluate the effectiveness of the proposed approach. Moreover, the proposed approach will be improved, such as combing domain adaptation analysis, to implement heterogeneous defect prediction.

Data Availability

The data used to support the findings of this study have been deposited in https://github.com/Sevensweett/Defect-Prediction.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The work was supported by the Scientific Research Foundation for the introduction of talent of Jiangsu University of Science and Technology, China; Natural Science Foundation of the Higher Education Institutions of Jiangsu Province, China (Grant No. 18JKB520011); and Primary Research and Development Plan (Social Development) of Zhenjiang City, China (Grant No. SH2019021).

References

[1] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, 2012.

[2] F. Rahman, S. Khatri, E. T. Barr, and P. Devanbu, "Comparing static bug finders and statistical prediction," in *Proceedings of the 36th International Conference on Software Engineering*, pp. 424–434, Hyderabad India, May 2014.

[3] X. Yang, K. Tang, and X. Yao, "A learning-to-rank approach to software defect prediction," *IEEE Transactions on Reliability*, vol. 64, no. 1, pp. 234–246, 2015.

[4] B. Ghotra, S. Mcintosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *Proceedings of the 37th IEEE International Conference on Software Engineering*, pp. 789–800, Florence, Italy, May 2015.

[5] W. Wen, B. Zhang, X. Gu, and X. Ju, "An empirical study on combining source selection and transfer learning for cross-project defect prediction," in *Proceedings of the 1st International Workshop on Intelligent Bug Fixing*, pp. 29–38, Hangzhou, China, February 2019.

[6] S. Tang, S. Huang, C. Zheng, E. Liu, C. Zong, and Y. Ding, "A novel cross-project software defect prediction algorithm based on transfer learning," *Tsinghua Science and Technology*, vol. 27, no. 1, pp. 41–57, 2022.

[7] J. Jiarpakdee, C. K. Tantithamthavorn, H. K. Dam, and J. Grundy, "An empirical study of model-agnostic techniques for defect prediction models," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 166–185, 2022.

[8] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2–13, 2007.

[9] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *Proceedings of the 30th International Conference on Software Engineering*, pp. 531–540, Leipzig, Germany, May 2008.

[10] A. E. Hassan, "Predicting faults using the complexity of code changes," in *Proceedings of the 31st International Conference on Software Engineering*, pp. 78–88, Vancouver, Canada, May 2009.

[11] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," *Automated Software Engineering*, vol. 19, no. 2, pp. 167–199, 2012.

[12] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," *Information and Software Technology*, vol. 54, no. 3, pp. 248–256, 2012.

[13] G. Canfora, A. D. Lucia, M. D. 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*, pp. 252–261, Luxembourg, March 2013.

[14] S. Herbold, A. Trautsch, and J. Grabowski, "A comparative study to benchmark cross-project defect prediction approaches," *IEEE Transactions on Software Engineering*, vol. 44, no. 9, pp. 811–833, 2018.

[15] 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, 2019.

[16] S. Watanabe, H. Kaiya, and K. Kaijiri, "Adapting a fault prediction model to allow inter language reuse," in *Proceedings of the 4th international workshop on Predictor models in software engineering*, pp. 19–24, Leipzig, Germany, May 2008.

[17] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, no. 5, pp. 540–578, 2009.

[18] S. Herbold, "Training data selection for cross-project defect prediction," in *Proceedings of the 9th International Conference on Predictive Models in Software Engineering*, pp. 1–10, Baltimore, USA, October 2013.

[19] T. Fukushima, Y. Kamei, S. Mcintosh, K. Yamashita, and N. Ubayashi, "An empirical study of just-in-time defect prediction using cross-project models," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, pp. 172–181, Hyderabad, India, May 2014.

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

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

[22] P. He, B. Li, D. Zhang, and Y. Ma, "Simplification of training data for cross-project defect prediction," *Computer Science*, pp. 1–17, arXiv:1405.0773, 2014.

[23] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proceedings of the 35th International Conference on Software Engineering*, pp. 382–391, San Francisco, USA, May 2013.

[24] C. Liu, D. Yang, X. Xia, M. Yan, and X. Zhang, "A two-phase transfer learning model for cross-project defect prediction," *Information and Software Technology*, vol. 107, pp. 125–136, 2019.

[25] S. Zheng, J. Gai, H. Yu, H. Zou, and S. Gao, "Training data selection for imbalanced cross-project defect prediction," *Computers & Electrical Engineering*, vol. 94, Article ID 107370.

[26] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: some comments on the nasa software defect datasets," *IEEE Transactions on Software Engineering*, vol. 39, no. 9, pp. 1208–1215, 2013.

[27] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, pp. 1–10, Timisoara, Romania, September 2010.

[28] P. W. Lamberti and A. P. Majtey, "Non-logarithmic jensen-shannon divergence," *Physica A: Statistical Mechanics and Its Applications*, vol. 329, no. 1-2, pp. 81–90, 2003.

[29] K. Fukunaga and L. Hostetler, "Optimization of k nearest neighbor density estimates," *IEEE Transactions on Information Theory*, vol. 19, no. 3, pp. 320–326, 1973.

[30] Y. P. Mack and M. Rosenblatt, "Multivariate k-nearest neighbor density estimates," *Journal of Multivariate Analysis*, vol. 9, no. 1, pp. 1–15, 1979.

[31] H. Yu, C. Sun, X. Yang, S. Zheng, and H. Zou, "Fuzzy support vector machine with relative density information for classifying imbalanced data," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 12, pp. 2353–2367, 2019.

[32] S. Zheng, J. Gai, H. Yu, H. Zou, and S. Gao, "Software defect prediction based on fuzzy weighted extreme learning machine with relative density information," *Scientific Programming*, vol. 2020, Article ID 8852705, 18 pages, 2020.

[33] Y. Zhou, Y. Yang, H. Lu et al., "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.

[34] X. Wan, Z. Zheng, and Y. Liu, "SPE^2: self-paced ensemble of ensembles for software defect prediction," *IEEE Transactions on Reliability*, vol. 71, no. 2, pp. 865–879, 2022.

[35] Z. Xu, S. Pang, T. Zhang et al., "Cross project defect prediction via balanced distribution adaptation based transfer learning," *Journal of Computer Science and Technology*, vol. 34, no. 5, pp. 1039–1062, 2019.

[36] S. Kim, E. J. Whitehead, and Y. Zhang, "Classifying software changes: clean or buggy?" *IEEE Transactions on Software Engineering*, vol. 34, no. 2, pp. 181–196, 2008.
[37] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin, "Liblinear: a library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.
[38] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pp. 2058–2065, Phoenix, Arizona, February 2015.
[39] X. Y. Jing, F. Wu, X. Dong, and B. Xu, "An improved sda based defect prediction framework for both within-project and cross-project class-imbalance problems," *IEEE Transactions on Software Engineering*, vol. 43, no. 4, pp. 321–339, 2017.
[40] L. Minku, F. Sarro, E. Mendes, and F. Ferrucci, "How to make best use of cross-company data for web effort estimation?" in *Proceedings of the 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 1–10, Beijing, China, November 2015.
[41] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.
[42] J. Demiar and D. Schuurmans, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1–30, 2006.
[43] J. Cohen, "Statistical power analysis for the behavioral sciences," *Journal of the American Statistical Association*, 2013.
[44] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," *Empirical Software Engineering*, vol. 17, no. 4-5, pp. 531–577, 2012.
[45] R. Wu, H. Zhang, S. Kim, and S. Cheung, "ReLink: recovering links between bugs and changes," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp. 15–25, Szeged, Hungary, September 2012.
[46] F. Zhang, I. Keivanloo, and Y. Zou, "Data transformation in cross-project defect prediction," *Empirical Software Engineering*, vol. 22, no. 6, pp. 3186–3218, 2017.
[47] 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.
[48] Q. Zou, L. Lu, S. Qiu, X. Gu, and Z. Cai, "Correlation feature and instance weights transfer learning for cross project software defect prediction," *IET Software*, vol. 15, no. 1, pp. 55–74, January 2021.
[49] 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, no. 2, Article ID 106588, 2021.
[50] C. Jin, "Cross-project software defect prediction based on domain adaptation learning and optimization," *Expert Systems with Applications*, vol. 171, no. 1, Article ID 114637, 2021.
[51] S. Mehta and K. S. Patnaik, "Improved prediction of software defects using ensemble machine learning techniques," *Neural Computing & Applications*, vol. 33, no. 16, pp. 10551–10562, 2021.