Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A software defect prediction method with metric compensation based on feature selection and transfer learning*

Jinfu CHEN[1,2], Xiaoli WANG[1,2], Saihua CAI[†‡1,2], Jiaping XU[1], Jingyi CHEN[1], Haibo CHEN[1]

[1]*School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang 212013, China*

[2]*Jiangsu Key Laboratory of Security Technology for Industrial Cyberspace, Jiangsu University, Zhenjiang 212013, China*

[†]E-mail: caisaih@ujs.edu.cn

Received Sept. 30, 2021; Revision accepted Feb. 5, 2022; Crosschecked Mar. 3, 2022; Published online Apr. 4, 2022

Abstract: Cross-project software defect prediction solves the problem of insufficient training data for traditional defect prediction, and overcomes the challenge of applying models learned from multiple different source projects to target project. At the same time, two new problems emerge: (1) too many irrelevant and redundant features in the model training process will affect the training efficiency and thus decrease the prediction accuracy of the model; (2) the distribution of metric values will vary greatly from project to project due to the development environment and other factors, resulting in lower prediction accuracy when the model achieves cross-project prediction. In the proposed method, the Pearson feature selection method is introduced to address data redundancy, and the metric compensation based transfer learning technique is used to address the problem of large differences in data distribution between the source project and target project. In this paper, we propose a software defect prediction method with metric compensation based on feature selection and transfer learning. The experimental results show that the model constructed with this method achieves better results on area under the receiver operating characteristic curve (AUC) value and F1-measure metric.

Key words: Defect prediction; Feature selection; Transfer learning; Metric compensation
https://doi.org/10.1631/FITEE.2100468 **CLC number:** TP311.5

1 Introduction

With the rapid development of information technology, software has been widely used in applications (Malhotra, 2015; Wahono, 2015). However, due to coding errors and incorrect understanding of software requirements, problems often occur in software (Chen X et al., 2018), and the defective software can cause incalculable losses once deployed. Software defect prediction is an important technique to ensure software quality and safety (Hall et al., 2012), which improves the quality of software products by identifying classes or modules with defects in advance to effectively allocate testing resources (Tsuda et al., 2019).

In the past two decades, many software defect prediction methods have been proposed to predict defects in a single project (called within-project defect prediction, or WPDP) (Wu et al., 2017), which trains the defect prediction model using a historical labeled dataset from the same project. The high

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 62172194 and U1836116), the National Key R&D Program of China (No. 2020YFB1005500), the Leading-edge Technology Program of Jiangsu Provincial Natural Science Foundation, China (No. BK20202001), the China Postdoctoral Science Foundation (No. 2021M691310), the Postdoctoral Science Foundation of Jiangsu Province, China (No. 2021K636C), and the Future Network Scientific Research Fund Project, China (No. FNSRFP-2021-YB-50)

ORCID: Jinfu CHEN, https://orcid.org/0000-0002-3124-5452; Saihua CAI, https://orcid.org/0000-0003-0743-1156

© Zhejiang University Press 2022

similarity and high coupling between modules in the same project allow the defect prediction model to achieve a good prediction performance. However, it is difficult to obtain enough historical data from the same project to train the model in practical applications, and because of the lack of historical data, the constructed model cannot achieve a high performance (Wan et al., 2020). A simple solution to this problem is to build a defect prediction model for the target project using high-quality datasets collected from other projects; this is called cross-project defect prediction (CPDP) (Ryu et al., 2017; Tabassum et al., 2020). CPDP is less accurate than WPDP (Liu et al., 2019) due to the distribution of metric values between the source and target project datasets (Iqbal et al., 2020). This problem has already attracted wide attention from researchers. For example, Watanabe et al. (2008) proposed a metric compensation method to improve the similarity between the source project and target project, but this method focuses only on using the source project to unilaterally adapt to the data distribution of the target project and do not consider the effect of redundant features on the efficiency of model training. In contrast, we would like to achieve bidirectional adaptation of the data distribution between the source project and target project. In addition, when the dimensionality of the project's features is too high, the spatial and temporal efficiency of model training affects the prediction accuracy of the model. If these two challenges can be solved, the defect prediction performance can be greatly improved. Our approach solves the bidirectional adaptation problem by combining metric compensation and transfer learning, and solves the dimensionality problem using the Pearson feature selection method. These solutions are the main contributions of this paper.

Based on the metric compensation technique, in this paper we propose a new method called peUpMeCom, that is, metric compensation based on transfer learning (Li et al., 2020b) and Pearson feature selection (Wang et al., 2010), to construct an efficient model for CPDP. peUpMeCom consists of two main components: feature selection (Saidi et al., 2019) and bidirectional metric compensation based on transfer learning (Rahman and Devanbu, 2013). In the model training phase, redundant features that are not related to the defect category are filtered out according to the Pearson coefficients (Yu et al., 2015; Shippey et al., 2019). The distribution difference of the features between the source project and target project is then reduced using a combination of transfer component analysis (TCA) and bidirectional metric compensation (Madeyski and Jureczko, 2015; Chen JY et al., 2019). The main contributions of this paper are as follows:

1. We propose a CPDP method (peUpMeCom) that combines Pearson feature selection and bidirectional metric compensation, which improves prediction accuracy of the model.

2. We introduce the Pearson feature selection method to filter redundant features that are unrelated to the defect category to improve the training efficiency of the model.

3. We use a combination of TCA and bidirectional metric compensation to reduce variability in the feature distribution of the source project and target project, thus improving the similarity between projects.

2 Related works

In recent years, many studies have been conducted in the field of software defect prediction (Chen JY et al., 2020). Due to limitations such as insufficient historical data, CPDP is also emerging as a popular research direction (Guo et al., 2018). CPDP trains prediction models using enough training data from existing source projects and then applies the model to the target project to obtain prediction results. Existing research focuses mainly on the following aspects of CPDP methods: reducing the differences in data distribution between the source project and target project, removing features that are irrelevant and redundant for model construction, and solving the class imbalance problem. These methods have also shown strong feasibility in experimental studies.

2.1 Reducing data distribution differences

The main study of CPDP was carried out by Briand et al. (2002), in which a prediction model was developed using linear regression. The localization of defect-prone classes was accurately predicted and such a model outperforms most irregular models in terms of class size. However, the predicted failure probabilities were not representative because of the variation between different projects.

To increase the distribution similarity of the features between the source project and target project, Nam et al. (2013) proposed a combination of normalization and TCA and then proposed the TCA+ method. This method first selects an optimal normalization method according to the target data. Then, TCA is applied to determine the mapping of the training and target data to the homogeneous metric space. The method can automatically select the best data normalization method after analyzing the characteristics of the source project and target project with the assistance of a set of pre-learned rules, so that the source project and target project have similar feature distributions.

In addition, Watanabe et al. (2008) proposed a metric compensation method. This method attempts to effectively improve the similarity between the source project and target project using the target set to adjust the data distribution of the source set (Li et al., 2020a). They analyzed the effect of CPDP between a Java project and C++ project. The metric compensation method was found to be effective in improving the accuracy and recall of CPDP, with a large increase in recall. Another improvement of the method was proposed; Herbold et al. (2018)'s idea was that it is better to let the training data adapt to the data distribution of the target set than to let the target set adapt to the data distribution of the source set during training. However, these two methods adjust only the data distribution in one direction, which lacks some flexibility. In addition, for projects with a large volume of data, these approaches consume too much time for model training, which reduces the model's prediction efficiency.

2.2 Deletion of redundant features

It has been shown that too many irrelevant and redundant features in a dataset can lead to degradation in the performance of the defect prediction model. He et al. (2015) proposed a solution from the viewpoint of feature selection that first analyzed different datasets and selected different feature subsets. Finally, the correlation between the features in the subsets was further analyzed, redundant features were removed, and minimized feature subsets were constructed in each dataset.

Similarly, with the transformation of the above idea, Amasaki et al. (2015) started with the target project and tried to remove irrelevant feature instances. Because the instances in the target projects were not labeled, they used unsupervised learning methods to identify these irrelevant feature instances to improve the prediction accuracy of the model.

2.3 Resolving class imbalances

Class imbalance problems are common in software defect datasets (Siers and Islam, 2015). There are more classes without defects than classes with defects, which results in low prediction accuracy of prediction models for defective classes. The currently available class imbalance methods (Peng et al., 2019) can be briefly divided into two categories: (1) cost-sensitive learning methods (McBride et al., 2019), a type of method that achieves class balance by setting different costs for different types of prediction errors; (2) sampling methods, including the random under sampling, random oversampling, and synthetic minority oversampling technique (SMOTE) methods (Purnami and Trapsilasiwi, 2017). Fukushima et al. (2014) and Kamei et al. (2016) reduced the number of majority class instances in the training set using the random under sampling method, so the number of majority class instances was consistent with the number of minority class instances to balance the defective classes. However, this method discards a large amount of defect-free data, which may cause deviation in the model prediction results. To solve this problem, Ryu et al. (2014, 2016) proposed the value-cognitive boosting with support vector machine (VCB-SVM) method. The method first calculates the similarity weight of each instance in the source project based on the dataset of the target project, and then constructs the defect prediction model using a boosting method based on an SVM.

After a comprehensive and holistic analysis of the advantages and disadvantages of the above proposed methods, we propose our method, i.e., peUpMeCom. Compared with traditional methods mentioned above, our method achieves better results, using metric compensation based on feature selection and transfer learning to resolve the differences between the source project and target project.

3 The proposed method

In this section, we propose a metric compensation method called peUpMeCom. peUpMeCom consists of two main phases: (1) Features are filtered

according to Pearson coefficients. Those features with a strong correlation with the defect category are selected, whereas redundant features with low or no correlation are removed. (2) Upgrade metric compensation (upMeCom) is used to reduce variance distribution of the source project and target project. upMeCom is a bidirectional metric compensation method based on transfer learning.

Fig. 1 shows the overall process of defect prediction of the proposed peUpMeCom method. As shown in Fig. 1, the first step in building the model is to collect feature instances from the source project. Because the datasets used in the experimental application are abstract numerical representations of projects, which include defect prediction metrics such as complexity metrics, module size metrics, program length, and the number of conditional statements, the values of these collected initial metrics are represented as initial feature instances in the study.

After the initial features of the source project have been collected, all instances of the feature set are traversed. The correlation coefficient is calculated between each feature and the defect category; then, feature correlation analysis is performed and feature redundancy is determined based on the magnitude of the correlation coefficient between the feature and the category. Too few features cannot fully represent the defective features of a project, and too many features can cause inefficiencies in model training. Before testing the test set, we select some data from each dataset to verify the appropriate number of features. Based on the Pearson correlation coefficient array, we record the area under the receiver operating characteristic curve (AUC) of the model with 10, 12, 15, 18, and 20 features. The experimental results are shown in Table 1, where we can see that the AUC for the test set peaks when the model includes 18 or more features. Therefore, the threshold value for the number of features is set to 18; that is, after obtaining the array of correlation coefficients in descending order, the features corresponding to the top 18 coefficients are selected to form the feature subset. Then, a bidirectional metric compensation method based on the transfer learning technique is applied to the projects. Initially, the project data is mapped to a latent feature space using TCA. Then, based on the low-dimensional features obtained from the prior process, the metric compensation technique is used to enhance the similarity between the source and target data distributions.

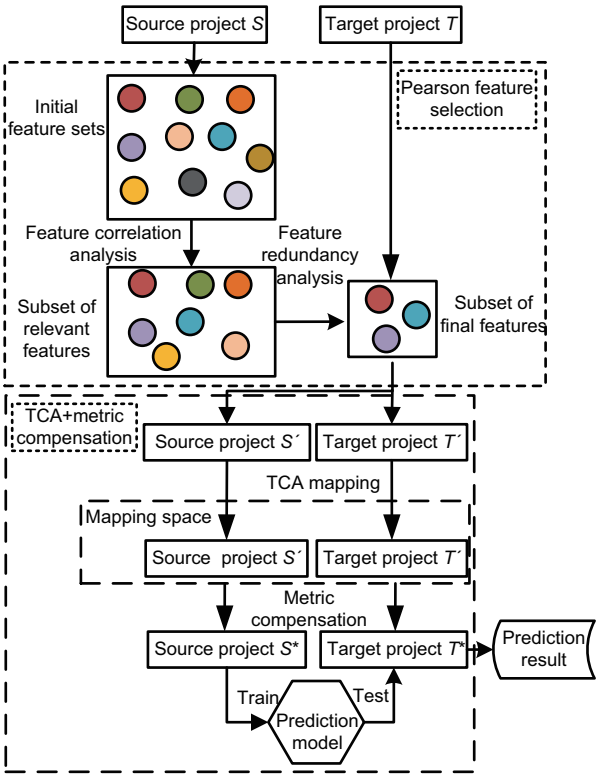

Fig. 1 Overall process of the proposed method for building a model

Table 1 AUC values for different number of features

Source=>target	AUC				
	Number=10	12	15	18	20
EQ=>JDT	0.592	0.599	0.615	**0.628**	0.623
EQ=>LC	0.614	0.627	0.657	**0.684**	0.669
JDT=>PDE	0.620	0.622	0.635	0.644	**0.645**
JDT=>ML	0.565	0.581	0.626	**0.638**	0.629
PDE=>ML	0.564	0.571	0.598	0.604	**0.605**
EQ=>ML	0.552	0.561	0.571	**0.573**	0.571
CM1=>JM1	0.537	0.543	0.565	0.610	**0.616**
KC1=>JM1	0.505	0.518	0.538	**0.577**	0.572
KC2=>JM1	0.507	0.540	0.572	**0.602**	0.595
PC1=>JM1	0.517	0.533	0.548	**0.572**	0.566
CM1=>KC1	0.581	0.598	0.623	0.644	**0.648**
KC2=>KC1	0.555	0.596	0.625	**0.658**	0.641
PC1=>KC1	0.599	0.606	0.614	0.642	**0.643**
CM1=>KC2	0.509	0.524	0.567	**0.655**	**0.655**
CM1=>PC1	0.558	0.563	0.587	**0.605**	**0.605**
KC2=>PC1	0.548	0.551	0.582	**0.649**	0.644
Safe=>Zxing	0.532	0.539	0.546	**0.578**	0.571
Safe=>Apache	0.525	0.530	0.556	**0.607**	0.596
Apache=>Safe	0.647	0.676	0.735	**0.825**	0.823
Apache=>Zxing	0.517	0.539	0.553	**0.587**	**0.587**
Zxing=>Safe	0.617	0.617	0.697	**0.705**	0.701
Zxing=>Apache	0.525	0.545	0.576	**0.637**	0.622

Best results are in bold

The algorithmic flow of the peUpMeCom method is shown in Algorithm 1, in which the feature selection phase is described at steps 1 and 2, the data transformation phase is described from steps 3 to 12, and the prediction phase ends from steps 13 to 15. First, the correlation coefficient between each feature and the defect category is calculated by Pearson correlation analysis, and then the final subset of features for training is selected based on the coefficient array R. After that, TCA is used to select the common characteristics of the source project and target project and map them to latent space φ. Then, the instance data in the transformed source project is traversed using a for loop. For each metric, a weight is assigned to the original metric value using the metric compensation method, so the compensated data value becomes more similar to the learned data, thus enhancing the resemblance between projects. The process will be described in detail next.

Algorithm 1 peUpMeCom

Input: Labeled source project S

Output: Unlabeled target project T

1: Define a coefficient correlation array $R=\{r_1, r_2, ..., r_m\}$
2: Define a feature subset F, $F=\text{set}_i$ if AUC(set_i) is maximum
3: Define $\boldsymbol{S}=[(s_{x_i}, y_{x_i})]_{i=1}^{n_1}$ and $\boldsymbol{T}=[t_{q_i}]_{i=1}^{n_2}$. Find a mapping space φ for $\boldsymbol{S}$ and $\boldsymbol{T}$, such that the value of $\text{Dist}(\varphi(\boldsymbol{S}), \varphi(\boldsymbol{T}))$ is minimized
4: Define the mapped source dataset $S'=\{s_1, s_2, ..., s_m\}$ and the target dataset $T'=\{t_1, t_2, ..., t_m\}$
5: **loop**
6: **for** j in metrics of S' **do**
7: **for** i in samples of T' **do**
8: $t'^j_i=\frac{t^j_i \times \text{mean}(s_j)}{\text{mean}(t_j)}$
9: $s'^j_i=\frac{s^j_i \times \text{mean}(t_j)}{\text{mean}(s_j)}$
10: **end for**
11: **end for**
12: **end loop**
13: For final source project S^* and target project T^*, feed S^* into a decision tree classifier for model training
14: Applying the model to T^*
15: **return** Prediction class label for T^*

For selection of machine learning classifiers, we use a decision tree classification model. Currently, there are several machine learning classification methods, and different classification techniques are applicable to different domains. The commonly used classifiers include naive Bayes (Habibi et al., 2018), SVM (Shuai et al., 2013), linear discriminant analysis (Lv, 2019), and decision tree (Marian et al., 2016). According to an experimental comparison of the defect prediction performances of these four classifiers based on six datasets, the decision tree classifier outperforms the other three classifiers in most cases, both in terms of PofB20 and F1-score metrics (Yang et al., 2017). In light of experimental experience, we choose the decision tree as the classifier for peUpMeCom in this study. In addition, the classification criterion for the defect category is determined by matching the features contained in the data to be tested with the path of the feature nodes of the constructed decision tree. The information represented by the leaf nodes in the matching path is the defect category information of the data to be tested. For entity data with a given unknown label, the prediction information of the entity category is obtained by traversing from the root node to the leaf nodes.

3.1 Pearson-based feature selection

Defect prediction datasets suffer commonly from the problem of dimensional catastrophe; the redundant and irrelevant features contained in the dataset can lead to decreased performance, excessive complexity, and extensive defect prediction model training time. Because feature selection can effectively alleviate this problem, it is necessary to select features that have a significant impact on the model before training it. There are two main types of feature selection methods: wrapper method and filter method (Thejas et al., 2021). The wrapper method treats the selection of subsets as a search and optimization process. By constantly comparing the combined effect of various feature subsets, the optimal feature subset is finally selected, but this method is prone to overfitting.

The filter method filters features based on weight order. For the defect prediction domain knowledge studied in this study, an initial indicator system needs to be established for relevance screening before predicting the software module, and the completed filtered indicator system is put into a machine learning classifier for training to produce results. The main relevance-based feature selection methods are Pearson coefficient, Spearman coefficient, and Kendall coefficient, which were described in detail in Grimm and Nesselroade (2018). Correlation coefficients are generally used to determine whether two variables are correlated and how closely they are related to each other. The Spearman and Kendall coefficients are generally appropriate for

variables of constant order. The Pearson coefficient is generally used to analyze the degree of correlation between characteristics and response variables. The module defect category is a constant-ratio variable that exhibits non-linear trend fluctuations, and there is some correlation between software module features and defect category predictions (i.e., there is variation in the extent to which software module features contribute, to defect category predictions). Cai et al. (2020) used Pearson coefficient to measure the correlation between module measures and module category linear correlation to filter out the impact. Therefore, in this study, Pearson correlation coefficients are used to filter the features; this method selects features that are strongly correlated with the defect class to form a subset of features for training the model. The main process of Pearson feature selection is shown in Fig. 2. A Pearson correlation coefficient is a linear correlation coefficient, and is used to reflect the degree of linear correlation between two variables, with the correlation coefficient r ranging from -1 to 1. The larger the absolute value of r, the stronger the correlation. Given a project dataset S, the number of samples and the feature dimensions are denoted by n and m, respectively. The j^{th} feature of the i^{th} sample is denoted as s_i^j. The class label of this sample is represented as y_i, and the correlation coefficient of the i^{th} feature with the defect class is calculated using Eq. (1), which is denoted by the correlation coefficient r:

$$r=\frac{\sum_{i=1}^{n}s_iy_i-\frac{1}{n}\sum_{i=1}^{n}s_i\sum_{i=1}^{n}y_i}{\sqrt{\sum_{i=1}^{n}s_i^2-\frac{1}{n}\left(\sum_{i=1}^{n}s_i\right)^2}\sqrt{\sum_{i=1}^{n}y_i^2-\frac{1}{n}\left(\sum_{i=1}^{n}y_i\right)^2}}, \tag{1}$$

where s_i represents the independent variable (e.g., a metric) and y_i represents the dependent variable (e.g., clean or buggy).

Through the calculation of the correlation coefficients of m features and defect categories, an array of coefficients, denoted by $R=\{r_1,r_2,...,r_m\}$, is constituted. Then, an ordered array of correlation coefficients is obtained, denoted by $R'=\{r_i',r_j',...,r_m'\}$, which is sorted by the values of elements in R from the largest to the smallest. The subscripts of the coefficients in R represent the subscripts of the features in the dataset. Then, the model trained by m feature subsets is experimented on the validation set, and the impact of m features on the prediction performance of the model is evaluated with the AUC, where the feature with the largest AUC is selected as the best feature subset for training the defect prediction model.

3.2 Metric compensation based on transfer learning

After obtaining the best subset of the features through feature selection, we combine TCA and

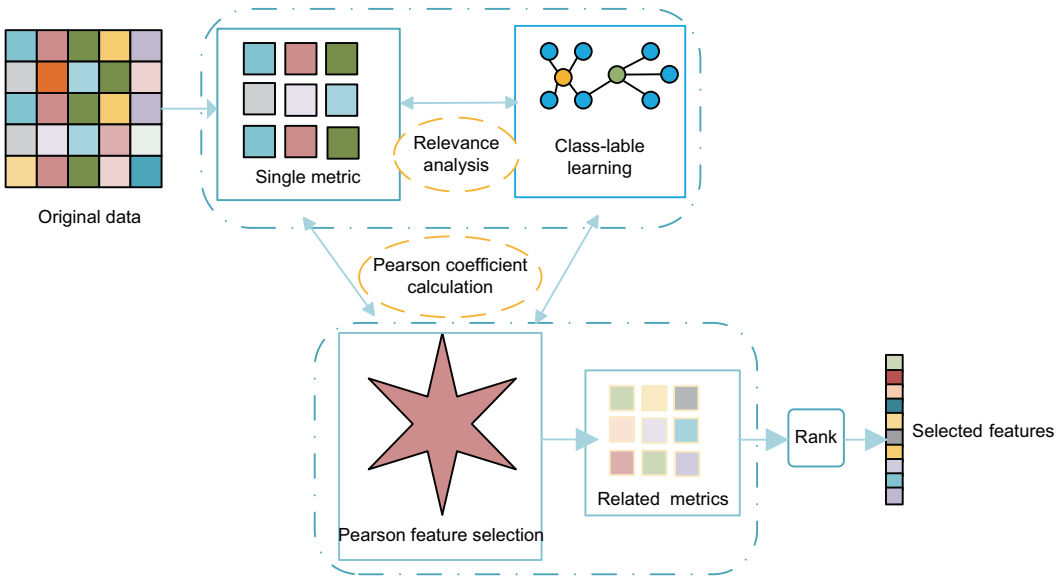

Fig. 2 Overall process of Pearson feature selection

bidirectional metric compensation techniques to further reduce the data distribution difference between the source project and target project. The overall process of metric compensation based on transfer learning is shown in Fig. 3. First, we use TCA to explore the potential common factors between the source project and target project, and then project them into the learned feature space to reduce the distribution differences of the data between the source domain and target domain, while preserving the original data structure. Assume that the data matrix of the source project is $\boldsymbol{S} = [(s_{x_i}, y_{x_i})]_{i=1}^{n_1}$, and that the data matrix of the target project after feature selection is $\boldsymbol{T} = [t_{q_i}]_{i=1}^{n_2}$, where n_1 and n_2 denote the numbers of samples of the source project and target project respectively, t_{q_i} denotes data instance information in the target dataset, s_{x_i} is the sample data information in the source project, and y_{x_i} denotes the defect information corresponding to this sample. The purpose of TCA is to attempt to learn a transformation φ that maps the original data from the source domain and target domain into a space where the differences between the domains are small and the variance is large. Assuming that transformation $\varphi : P^m \to P^n$ maps the original m-dimensional feature vector into an n-dimensional subspace, the objective of TCA can be expressed as

$$\begin{aligned} &\arg\min \{\mathrm{Dist}(\varphi(\boldsymbol{S}), \varphi(\boldsymbol{T})) + \lambda P(\varphi)\} \\ &\text{s.t. constraints on } \varphi(\boldsymbol{S}) \text{ and } \varphi(\boldsymbol{T}), \end{aligned} \tag{2}$$

where $\mathrm{Dist}(\varphi(\boldsymbol{S}), \varphi(\boldsymbol{T}))$ is measured by the maximum mean difference (MMD), $P(\varphi)$ is a regularization term to prevent overfitting, and $\lambda \geq 0$ is a trade-off parameter to control the effects of regularization. MMD is calculated as follows:

$$\mathrm{MMD} = \left\| \frac{1}{n_1} \sum_{i=1}^{n_1} \varphi(s_{x_i}) - \frac{1}{n_2} \sum_{i=1}^{n_2} \varphi(t_{q_i}) \right\|^2. \tag{3}$$

Transformation expressions can be expressed as $\varphi(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{\Omega}$, where $\boldsymbol{\Omega} \in P^{m \times n}$ is a matrix that maps m-dimensional feature vectors to n-dimensional feature vectors. After learning the transformation using TCA, the transformed data can be represented by $\varphi(\boldsymbol{S}) = \boldsymbol{S}\boldsymbol{\Omega}$ and $\varphi(\boldsymbol{T}) = \boldsymbol{T}\boldsymbol{\Omega}$.

In the space mapped by TCA, the source project data is represented as $S' = \{s_1, s_2, ..., s_m\}$, and the target project is represented as $T' = \{t_1, t_2, ..., t_n\}$. Subsequently, we apply a bidirectional metric compensation technique to the processed source and target project data. Denoting the j^{th} metric in the i^{th} sample data of the source project as s_i^j, we assign a weight to each metric in the source project to make the transformed metric more consistent with the distribution of the target data. The j^{th} metric mentioned here is not sorted according to a certain method, but refers to the default sort order of the metric elements in the downloaded project dataset. Similarly, we adopt the idea of weighting for the target project metrics to make the transformed target data fit the source project data after training. The weight is set to the ratio of the mean value of the specified metric of the source project to that of the target project. The two-way conversion can improve the training efficiency and accuracy of the model. The transformed source project metric is denoted by s'^j_i, and the target project metric is denoted by t'^j_i.

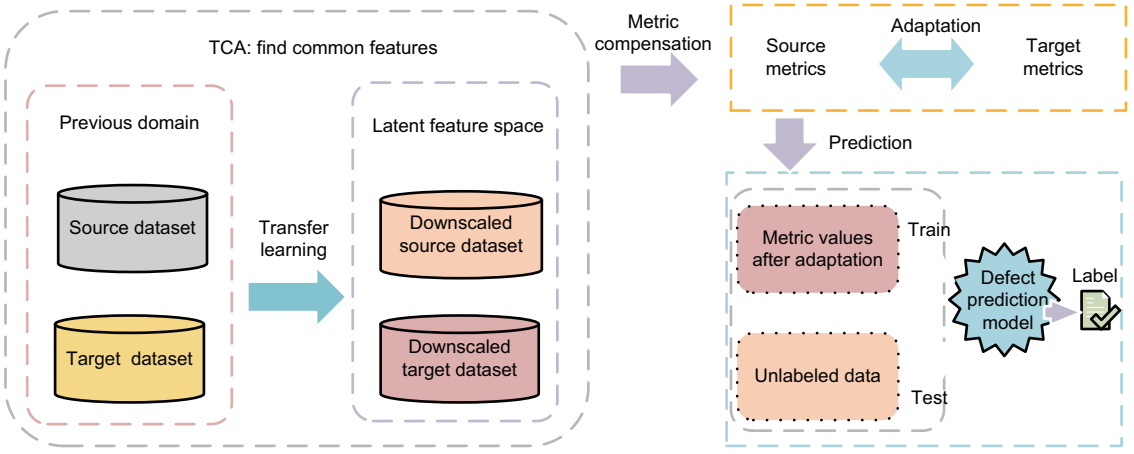

Fig. 3 Overall process of metric compensation based on transfer learning

The specific formulas used in the metric compensation process are as follows:

$$t'^{j}_{i} = \frac{t^{j}_{i} \times \text{mean}(s_j)}{\text{mean}(t_j)}, \tag{4}$$

$$s'^{j}_{i} = \frac{s^{j}_{i} \times \text{mean}(t_j)}{\text{mean}(s_j)}, \tag{5}$$

where $\text{mean}(s_j)$ denotes the mean value of the j^{th} metric in all samples of the source project, and $\text{mean}(t_j)$ denotes the mean value of the j^{th} metric in all sample data of the target project. Finally, the processed source project data is fed into a decision tree classifier for training the model, and the trained model is then used to predict the defects of the target data.

4 Experiments

We conducted several experiments to verify that the combination of feature selection and bidirectional metric compensation based on transfer learning can improve the performance of the defect prediction model. We used the NASA metric data program (MDP), Relink, and AEEEM datasets in these experiments. Each instance in the dataset represents a module, and the granularity of the module representation is different for different datasets; for the NASA MDP dataset, the module granularity is method, while for AEEEM, the module granularity is class, and for the Relink dataset, the module granularity is file. Dataset details are presented in Table 2.

Table 2 Details of the selected datasets

Dataset	Project	Number of instances	Percentage of buggy instances	Number of metrics
NASA MDP	JM1	10 885	19.3%	21
	KC1	2109	15.4%	21
	KC2	522	20.4%	21
	PC1	1109	6.94%	21
	CM1	498	9.84%	21
AEEEM	EQ	324	39.8%	61
	JDT	997	20.7%	61
	LC	691	9.3%	61
	ML	1862	13.2%	61
	PDE	1292	16.1%	61
Relink	Apache	194	50.5%	26
	Safe	56	39.3%	26
	Zxing	399	29.6%	26

4.1 Dataset description

The NASA MDP repository has been widely used for software defect prediction. We selected five projects from this repository that have the same metrics in our experiments. Each project represents a NASA software system that contains the corresponding defect marking data and various static code metrics. The static code metrics of the NASA MDP repository include size, complexity, etc., which are closely related to software quality. Because our approach was carried out in homogeneous CPDP, we chose five datasets that share the same 21 metric structures in the NASA MDP dataset for the experiments. This dataset is large; therefore, to verify the validity of the method, for selecting the source and target project pairs for subsequent experiments, we used random matching to select 10 data pairs for experiments.

AEEEM is a collection of apache lucene (LC), equinox (EQ), eclipse JDT core (JDT), eclipse PDE UI (PDE), and Mylyn (ML) projects. Each dataset consists of 61 metrics: 17 source code metrics, 5 prior defect metrics, 5 change entropy metrics, 17 source code entropy metrics, and 17 source code churn metrics.

The Relink dataset includes 26 metrics, all of which focus on code complexity. The items analyzed were Apache HTTP Server, Safe, and Zxing, which are widely used in defect prediction. We can see that with only 56 instances in the Safe dataset, it is difficult to support the model training. We chose this dataset because it could be compared with the experimental data obtained from the TCA method. In addition, to address the problem of small sample of data instances in the Safe dataset, the SMOTE oversampling method was used to expand the sample in our experiments to mitigate the problem appropriately.

4.2 Baseline comparison

To evaluate the performance of the proposed peUpMeCom method, three methods, that is, meCom (the original metric compensation) (Watanabe et al., 2008), meCom17 (the improved metric compensation) (Herbold et al., 2018), and the TCA method (Nam et al., 2013), were used as the baselines.

1. meCom (original metric compensation) (Watanabe et al., 2008)

meCom is an effective CPDP method that improves the homogeneity between the training data

and target data by adjusting the target data to ensure that it fits the distribution of the source data. Because the averages of each metric value are different between different projects, the authors thought that these values needed to be normalized. They considered the following compensation method: Supposing that error-prone files of project A are predicted using the training data of project B, then the test data will be compensated as follows:

$$\hat{m}_i(s^*) = \frac{m_i(s^*) \times \text{mean}(m_i(S))}{\text{mean}(m_i(S^*))}, \tag{6}$$

where $\hat{m}_i(s^*)$ denotes the s^{th} metric value of the i^{th} instance in project A after compensation, $m_i(s^*)$ denotes the original metric value, $\text{mean}(m_i(S))$ denotes the mean value of the s^{th} metric of all instances in project A, and $\text{mean}(m_i(S^*))$ denotes the mean value of the s^{th} metric of all instances in project B.

2. meCom17 (improved metric compensation) (Herbold et al., 2018)

meCom17 is an outstanding CPDP method that improves the homogeneity between the training data and target data by standardizing the training data instead of the target data. The authors' idea is that instead of using the target set to adapt to the data distribution of the training set, it is better to adapt to the data distribution of the target set at the time of training. The target data will be compensated as follows:

$$\hat{m}_i(s) = \frac{m_i(s) \times \text{mean}(m_i(S^*))}{\text{mean}(m_i(S))}, \tag{7}$$

where $\hat{m}_i(s)$ denotes the s^{th} metric value of the i^{th} instance in project B after compensation and $m_i(s)$ denotes the original metric value.

3. TCA

The main motivation for TCA is that there may be some common underlying factors between the source domain and target domain, although the characteristics of the observed domains are different. To reveal the underlying factors, the domains are projected onto a new space, which is called the potential space. In this way, the differences between domains can be reduced and the original data structure can be preserved. In other words, TCA attempts to learn a transformation φ to map the original data in the source domain and target domain to a potential space, where the differences between different domains are small and the variance of the transformed data is large. Before applying TCA, we apply min-max normalization to pre-process the data which is commonly used in machine learning literature. Normalization is performed so that all features of the dataset have the same weight and it is useful for classification.

4.3 Evaluation of prediction models

In this study, AUC and F1-measure are used as the evaluation criteria. A good prediction model should have high precision and high recall, but high recall is often achieved at the cost of low precision. Consequently, we use F1-measure to evaluate the performance, which is used to measure the harmonic average of the precision and recall. F1-measure and AUC are defined as

$$\text{F1-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{8}$$

$$\text{AUC} = \frac{\sum_{i \in \text{positiveClass}} \text{rank}_i - \frac{M(1+M)}{2}}{M \times N}, \tag{9}$$

where precision is defined as TP/(TP+FN) and recall is defined as FP/(FP+TN). Herein, TP, TN, FP, and FN are defined in Table 3. rank_i represents the ordinal number of the i^{th} sample (the probability score is ranked from the smallest to the largest, in the rank position) and M and N are the numbers of positive and negative samples respectively; $\sum_{i \in \text{positiveClass}} \text{rank}_i$ indicates that only the ordinal numbers of positive samples are summed.

Table 3 Defect prediction results

	Number of instances	
	Defective	Non-defective
Predicted as defective	TP	FP
Predicted as non-defective	FN	TN

4.4 Experimental results

4.4.1 RQ1: How well does our approach perform?

Scheme 1 To verify the effectiveness of the proposed peUpMeCom method, we compared it with three baseline methods based on the NASA MDP, Relink, and AEEEM datasets. Taking the AEEEM dataset as an example, when EQ is selected as the source project and JDT is selected as the target project, it can be expressed as EQ=>JDT, where

the left side of "=>" indicates the source project and the right side of "=>" indicates the target project. In addition, to verify that the advantages of the peUpMeCom method are fully demonstrated, for the same set of source dataset and target dataset, we conducted 500 experiments for all methods. The average values of all experiments based on the AEEEM, Relink, and NASA MDP datasets were calculated as the final experimental results, which are shown in Tables 4, 5, and 6, respectively. To statistically analyze the detailed prediction results of the dataset, we also used a statistical test, that is, the Wilcoxon rank sum test. The confidence level in the experiments was selected as 95%. The specific results in terms of AUC and F1-measure of the statistical tests are reported in Tables 7 and 8, respectively. In addition, we drew the experimental result of each experiment in box plots to analyze the stability of the proposed model.

Finding 1 Tables 4–6 show the experimental results of meCom, meCom17, TCA, and peUpMeCom based on the selected datasets. These tables show that in most cases, our method presents a clear advantage based on the experimental datasets, in terms of both AUC and F1-measure, and outperforms the meCom, meCom17, and TCA methods. Taking the AUC metric as an example, for EQ=>JDT based on the AEEEM dataset, peUpMeCom improves the AUC value by 0.023 compared to meCom, by 0.025 compared to meCom17,

Table 4 Comparison results among different methods based on the AEEEM dataset

Source=>target	AUC				F1-measure			
	meCom	meCom17	TCA	peUpMeCom	meCom	meCom17	TCA	peUpMeCom
EQ=>JDT	0.663	0.661	0.572	**0.686**	0.658	0.655	0.410	**0.684**
EQ=>LC	0.665	0.664	0.565	**0.688**	0.682	0.681	0.231	**0.703**
JDT=>PDE	0.647	0.647	0.548	**0.661**	0.616	0.618	0.278	**0.633**
JDT=>ML	0.626	0.626	0.530	**0.635**	0.525	0.523	0.240	**0.541**
PDE=>ML	0.606	0.605	0.534	**0.623**	0.612	0.613	0.226	**0.625**
EQ=>ML	0.594	0.594	0.522	**0.609**	0.560	0.561	0.241	**0.597**

Best results are in bold

Table 5 Comparison results among different methods based on the Relink dataset

Source=>target	AUC				F1-measure			
	meCom	meCom17	TCA	peUpMeCom	meCom	meCom17	TCA	peUpMeCom
Safe=>Apache	0.633	0.634	0.435	**0.656**	0.587	0.590	0.511	**0.628**
Safe=>Zxing	0.553	0.555	0.470	**0.581**	0.411	0.413	0.387	**0.539**
Apache=>Safe	0.649	0.655	0.389	**0.780**	0.591	0.595	0.413	**0.777**
Apache=>Zxing	0.566	0.563	0.528	**0.610**	0.427	0.424	0.362	**0.585**
Zxing=>Safe	0.612	0.613	0.384	**0.678**	0.482	0.486	0.372	**0.674**
Zxing=>Apache	0.588	0.584	0.551	**0.634**	0.487	0.481	0.480	**0.601**

Best results are in bold

Table 6 Comparison results among different methods based on the NASA MDP dataset

Source=>target	AUC				F1-measure			
	meCom	meCom17	TCA	peUpMeCom	meCom	meCom17	TCA	peUpMeCom
CM1=>JM1	0.607	0.606	0.522	**0.625**	0.513	0.516	0.309	**0.556**
KC1=>JM1	0.594	0.592	0.536	**0.613**	0.549	0.552	0.311	**0.575**
KC2=>JM1	0.608	0.609	0.567	**0.628**	0.548	0.544	0.349	**0.586**
PC1=>JM1	0.598	0.598	0.523	**0.614**	0.469	0.463	0.239	**0.518**
CM1=>KC1	0.645	0.647	0.525	**0.665**	0.582	0.599	0.275	**0.639**
KC2=>KC1	0.687	0.677	0.598	**0.697**	0.672	0.634	0.403	**0.678**
PC1=>KC1	0.653	0.654	0.501	**0.676**	0.589	0.596	0.202	**0.643**
CM1=>KC2	0.661	0.667	0.524	**0.702**	0.572	0.583	0.314	**0.648**
CM1=>PC1	0.616	0.621	0.549	**0.646**	0.542	0.560	0.255	**0.605**
KC2=>PC1	0.648	0.622	0.588	**0.653**	0.613	0.573	0.264	**0.621**

Best results are in bold

Table 7 Statistical test results between meCom and peUpMeCom in terms of AUC

Source=>target	p-value	Effect size
EQ=>JDT	2.16×10^{-123}	1.41
EQ=>LC	5.82×10^{-82}	1.32
JDT=>PDE	5.94×10^{-89}	1.43
JDT=>ML	5.88×10^{-85}	1.64
PDE=>ML	1.55×10^{-82}	1.33
EQ =>ML	2.69×10^{-75}	1.21
CM1=>JM1	2.38×10^{-43}	1.05
KC1=>JM1	2.72×10^{-45}	1.28
KC2=>JM1	5.20×10^{-52}	1.64
PC1=>JM1	1.61×10^{-38}	1.55
CM1=>KC1	7.79×10^{-51}	1.48
KC2=>KC1	2.21×10^{-14}	0.73
PC1=>KC1	2.87×10^{-37}	1.06
CM1=>KC2	3.34×10^{-69}	1.55
CM1=>PC1	4.52×10^{-10}	1.52
KC2=>PC1	0.006	0.55
Safe=>Apache	2.98×10^{-10}	0.80
Safe=>Zxing	2.06×10^{-17}	0.84
Apache=>Safe	5.61×10^{-159}	2.36
Apache=>Zxing	1.36×10^{-33}	1.39
Zxing=>Safe	8.19×10^{-26}	1.35
Zxing=>Apache	2.18×10^{-26}	0.91

Table 8 Statistical test results between meCom and peUpMeCom in terms of F1-measure

Source=>target	p-value	Effect size
EQ=>JDT	1.71×10^{-56}	0.73
EQ=>LC	5.45×10^{-46}	0.60
JDT=>PDE	3.24×10^{-43}	1.21
JDT=>ML	9.08×10^{-15}	0.76
PDE=>ML	3.38×10^{-11}	0.54
EQ =>ML	3.19×10^{-26}	0.70
CM1=>JM1	5.30×10^{-28}	0.71
KC1=>JM1	6.97×10^{-9}	0.53
KC2=>JM1	1.44×10^{-22}	1.28
PC1=> JM1	4.61×10^{-24}	1.07
CM1=> KC1	1.22×10^{-45}	1.19
KC2=>KC1	1.13×10^{-7}	0.52
PC1=>KC1	8.05×10^{-31}	1.01
CM1=> KC2	4.64×10^{-56}	1.28
CM1=> PC1	1.20×10^{-6}	1.18
KC2=>PC1	0.0483	0.6
Safe=>Apache	3.53×10^{-9}	0.64
Safe=>Zxing	4.22×10^{-62}	1.60
Apache =>Safe	6.43×10^{-146}	2.61
Apache=>Zxing	7.21×10^{-86}	3.16
Zxing=>Safe	7.11×10^{-71}	2.48
Zxing=>Apache	1.32×10^{-38}	1.36

and by 0.114 compared to TCA. For Apache=>Safe based on the Relink dataset, there is an improvement of 0.131 over meCom, 0.125 over meCom17, and 0.391 over TCA. For the NASA MDP dataset with CM1=>KC2, our method represents an improvement of 0.041 compared to meCom, 0.035 compared to meCom17, and 0.178 compared to TCA. These results show that our method constitutes a better prediction model for defect prediction when dealing with differences in data distribution across different domains compared to the baseline methods. This is attributed to the feature selection and TCA dimensionality reduction techniques. Pearson feature selection is used to eliminate features that are not relevant to the defect category prediction, while the metric compensation technique based on transfer learning ensures the similarity of the data distribution between the source project and target project in a low-dimensional space, which leads to an increase in the prediction accuracy of the model.

From the perspective of statistical test results, we can see from Tables 7 and 8 that all the p-values are less than 0.05 and that all the effect-size values are more than 0.5, confirming the significant difference between our method and the baseline method. Figs. 4, 5, and 6 show the prediction performances in terms of AUC and F1-measure metrics of different methods based on the NASA MDP, AEEEM, and Relink datasets, respectively. Most of the predicted values of the peUpMeCom method are higher than those of the baseline methods, and there are almost no outliers in the experimental data; most of the data resides in the box of the box-line plot, which implies high stability of the proposed method for model prediction performance.

4.4.2 RQ2: How does Pearson feature selection affect the prediction performance of the model?

Scheme 2 To evaluate the impact of the Pearson feature selection method on the model prediction performance, we compared a model that was trained using full metrics with a model that was trained using the optimal subset of features selected using the feature selection method, and conducted experiments based on the NASA MDP, Relink, and AEEEM datasets. The specific results of the prediction examples based on the AEEEM, NASA MDP, and Relink datasets are shown in Tables 9, 10, and 11, respectively. For all projects, to make the experimental data more convincing, experiments using the same dataset maintained the same experimental environment. In addition, we plotted the results for each set of experiments as box plots to verify the stability of the method for model prediction.

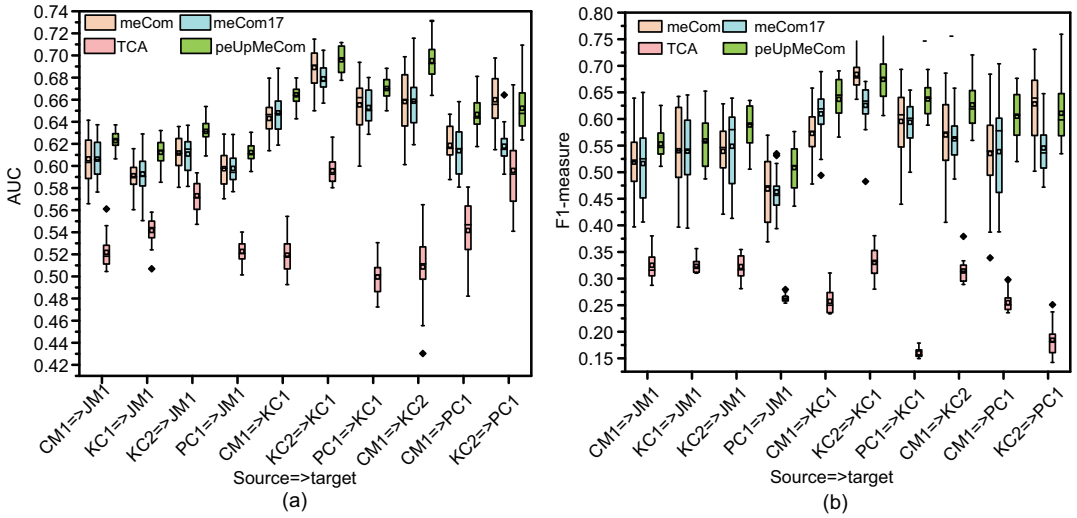

Fig. 4 Performance plots of different techniques based on the NASA MDP dataset: (a) AUC; (b) F1-measure

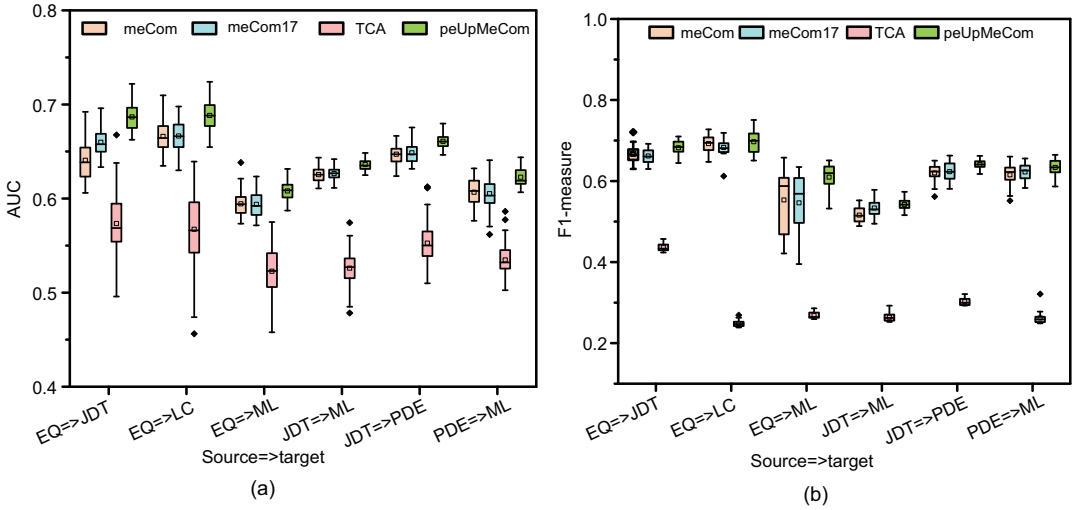

Fig. 5 Performance plots of different techniques based on the AEEEM dataset: (a) AUC; (b) F1-measure

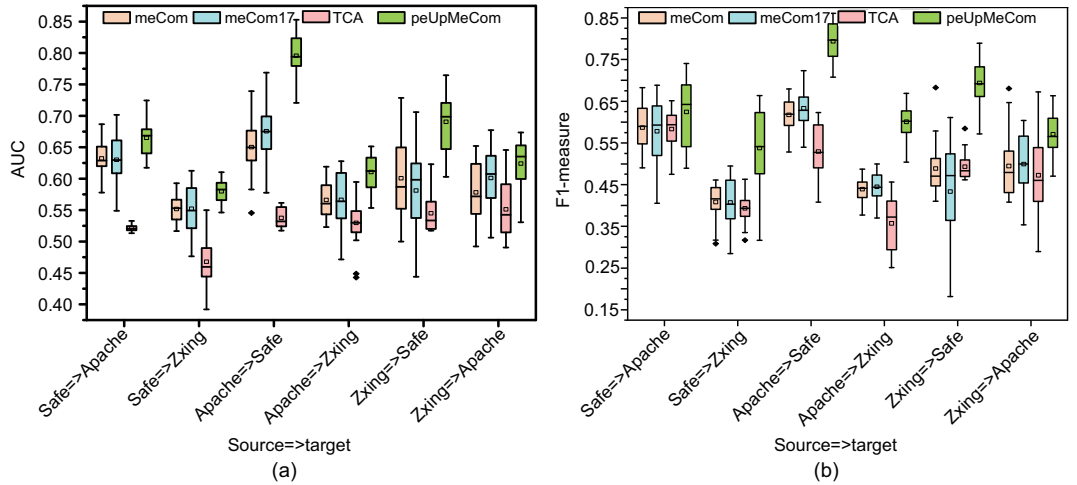

Fig. 6 Performance plots of different techniques based on the Relink dataset: (a) AUC; (b) F1-measure

Finding 2 Figs. 7, 8, and 9 show the prediction results of the feature selection method based on the NASA MDP, AEEEM, and Relink datasets respectively. Tables 9, 10, and 11 show the AUC and F1-measure values that are specific to the predicted cases, based on the AEEEM, NASA MDP, and Relink datasets respectively, with larger F1-measure and AUC values in bold in the tables. For the NASA MDP, Relink, and AEEEM datasets, in most cases (not generalized to 100% cases), the experimental results after feature selection are better than the original results. Taking EQ=>JDT in Table 9 for example, the original AUC value without feature selection is 0.639, and the AUC value after using the Pearson feature selection method is 0.686; the experimental result is 7.35% better than the original one, and similarly, the F1-measure is increased by 4.75%. It is obvious that the Pearson feature selection method has improved the prediction efficiency of the model. The main reason is that deleting redundant features that are unrelated to the defect category effectively reduces the model training time and weakens the interference of redundant features on the model prediction, thus improving the prediction performance and accuracy of the model.

Table 9 Results of the original data and Pearson feature selection based on the AEEEM dataset

Source=>target	AUC		F1-measure	
	Original	Pearson	Original	Pearson
EQ=>JDT	0.639	**0.686**	0.653	**0.684**
EQ=>LC	0.657	**0.688**	0.673	**0.703**
JDT=>PDE	0.646	**0.661**	0.615	**0.633**
JDT=>ML	0.605	**0.635**	**0.596**	0.541
PDE=>ML	0.596	**0.623**	0.587	**0.625**
EQ=>ML	0.582	**0.609**	0.594	**0.597**

Best results are in bold

Table 10 Results of the original data and Pearson feature selection based on the NASA MDP dataset

Source=>target	AUC		F1-measure	
	Original	Pearson	Original	Pearson
CM1=>JM1	0.570	**0.625**	0.453	**0.556**
KC1=>JM1	0.545	**0.613**	0.517	**0.575**
KC2=>JM1	0.585	**0.628**	0.501	**0.586**
PC1=>JM1	0.564	**0.614**	0.405	**0.518**
CM1=>KC1	0.637	**0.665**	0.557	**0.639**
KC2=>KC1	0.617	**0.697**	0.554	**0.678**
PC1=>KC1	0.636	**0.676**	0.547	**0.643**
CM1=>KC2	0.676	**0.702**	0.608	**0.648**
CM1=>PC1	0.609	**0.646**	0.517	**0.605**
KC2=>PC1	0.621	**0.653**	0.563	**0.621**

Best results are in bold

Table 11 Results of the original data and Pearson feature selection based on the Relink dataset

Source=>target	AUC		F1-measure	
	Original	Pearson	Original	Pearson
Safe=>Apache	0.633	**0.656**	0.589	**0.628**
Safe=>Zxing	0.556	**0.581**	0.413	**0.539**
Apache=>Safe	0.651	**0.780**	0.593	**0.777**
Apache=>Zxing	0.568	**0.610**	0.432	**0.585**
Zxing=>Safe	0.616	**0.678**	0.487	**0.674**
Zxing=>Apache	0.590	**0.634**	0.494	**0.601**

Best results are in bold

4.4.3 RQ3: Does the transfer learning technique improve metric compensation?

Scheme 3 To validate the contribution of transfer learning to the bidirectional metric compensation technique in model prediction performance, we conducted experiments based on these three datasets using different metric compensation methods. To demonstrate the effectiveness of the experiments, two baseline methods were chosen for comparison: meCom and meCom17. One project was then selected as the target project, and the remaining projects were used as the source datasets for training the model. As an example, when CM1 is used as the test set, JM1, KC1, KC2, and PC1 are used together as the training set for model training. Five hundred experiments were conducted on the same project of experimental data, and the average of the experimental values was taken as the final result for this target dataset; the results are shown in Table 12. In addition, to confirm the stability of the method, we plotted the prediction results of each set of experiments as box plots, and the comparison of the experimental results of the same set under different methods is shown in Fig. 10.

Finding 3 Fig. 10 shows the AUC values for CPDP based on the three datasets for the method combining transfer learning and bidirectional metric compensation compared to the traditional metric compensation methods. Table 12 shows the average multiple-to-one CPDP results for several project prediction cases. From Table 12 and Fig. 10, we can see that the model combining transfer learning and bidirectional metric compensation achieves a higher AUC than the baselines in most cases. Taking the KC1 target set as an example ((CM1+JM1+KC2+PC1)=>KC1), upMeCom can improve the average AUC value by 0.05 (0.628–

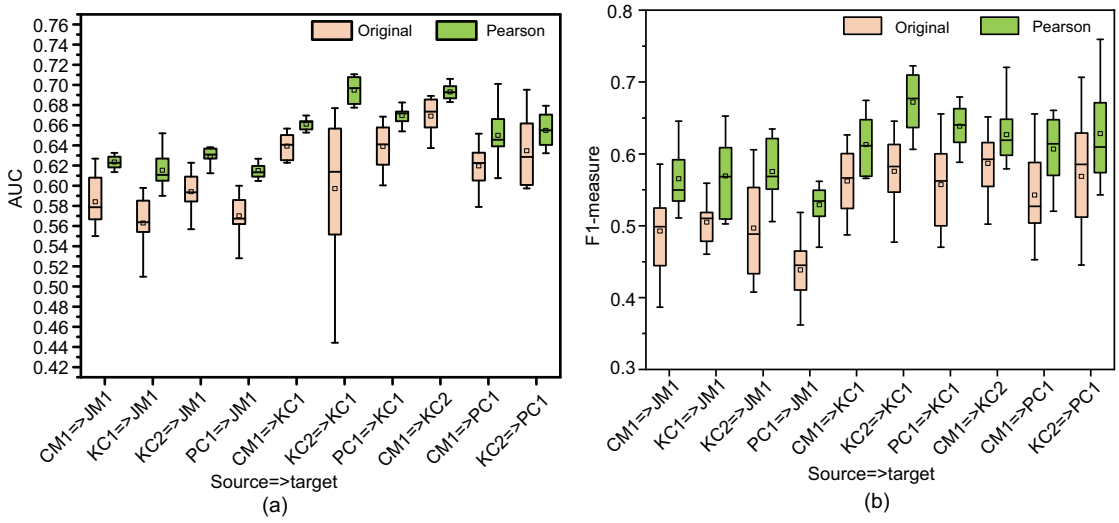

Fig. 7 Performance plots of feature selection based on the NASA MDP dataset: (a) AUC; (b) F1-measure

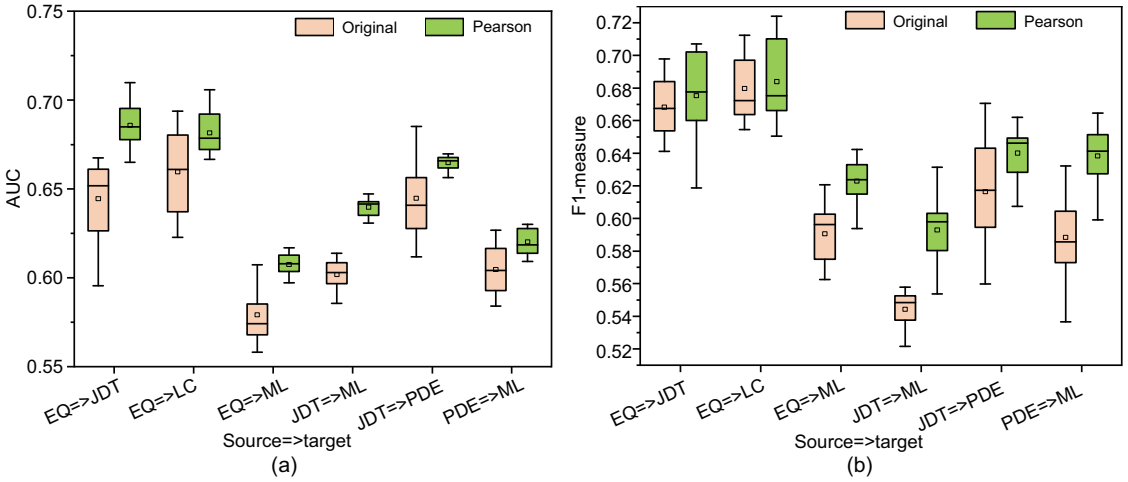

Fig. 8 Performance plots of feature selection based on the AEEEM dataset: (a) AUC; (b) F1-measure

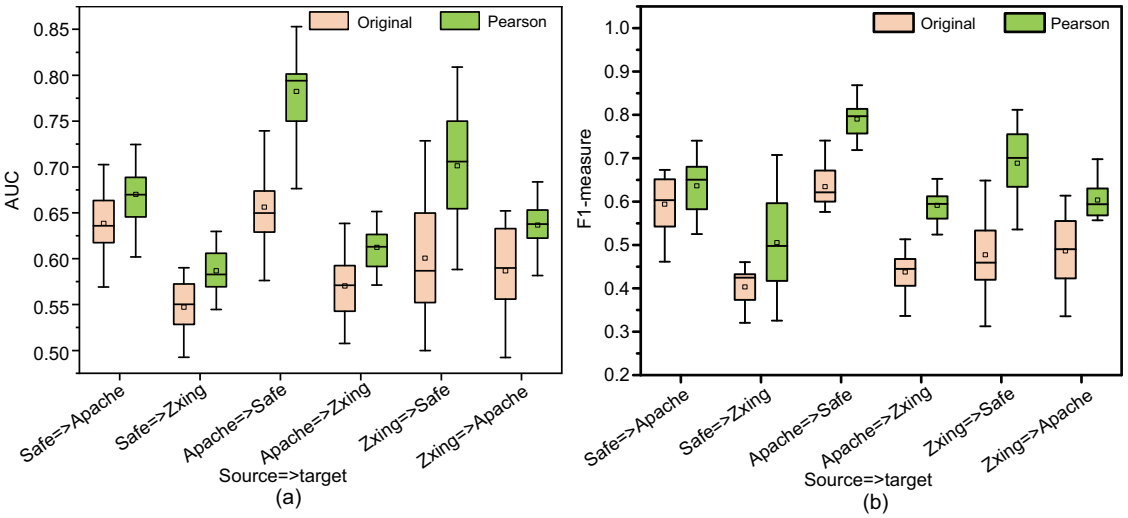

Fig. 9 Performance plots of feature selection based on the Relink dataset: (a) AUC; (b) F1-measure

0.578) compared to meCom and by 0.085 (0.628–0.543) compared to meCom17. This indicates that the combination of transfer learning and metric compensation can effectively improve the model prediction performance. One reason for this performance improvement is that the fusion of the TCA technique allows the training and testing samples to retain the maximum attribute features of the data in the low-dimensional feature space. Another reason is that traditional metric compensation methods tend to transform the data in one direction, whereas our bidirectional metric compensation combines the idea of bidirectional metric value conversion with a further reduction of the variability of the source and target project feature distributions. In addition, it can be seen from the box-line plots in Fig. 10 that the experimental data derived from our method is basically outlier-free, which indicates that the model constructed by this method is relatively stable.

Table 12 AUC obtained from the upMeCom and other metric compensation methods

Target project	AUC		
	meCom	meCom17	upMeCom
CM1	0.573	0.569	**0.612**
JM1	0.555	0.542	**0.561**
KC1	0.578	0.543	**0.628**
KC2	0.609	0.601	**0.671**
PC1	0.599	0.601	**0.615**
EQ	0.664	0.680	**0.695**
JDT	0.597	0.619	**0.644**
LC	0.631	0.642	**0.671**
ML	0.582	0.588	**0.608**
PDE	0.591	0.588	**0.618**
Safe	0.685	0.683	**0.730**
Zxing	0.588	0.572	**0.594**
Apache	0.619	0.612	**0.681**

Best results are in bold

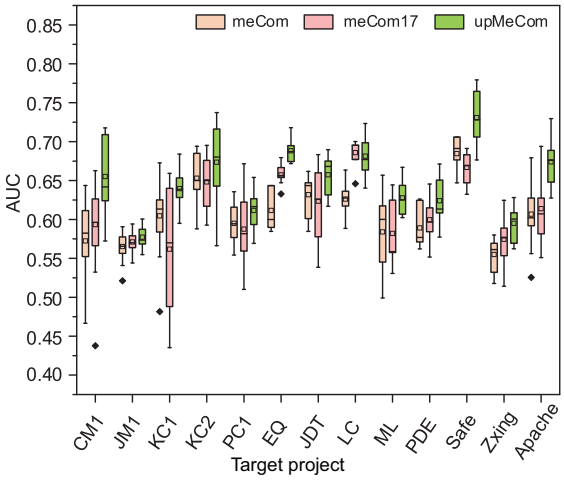

Fig. 10 AUC values under different metric compensation methods

5 Discussions

5.1 Insights learned from our empirical study

Our empirical study reveals some important facts that provide new insights to further advance CPDP research. Our comprehensive experimental results show that the metric compensation technique combining feature selection and transfer learning can improve CPDP performance. In particular, this technique can shorten model training time and effectively improve prediction accuracy of the model. Such findings provide important insight and guidance for future research: when predicting defects, if the data dimensionality is too large, the ideas of feature dimensionality reduction and enhanced data distribution similarity can be fused to provide new research directions for the development of CPDP techniques.

5.2 Threats to validity

Two potential threats to the effectiveness of the experiment are presented here. Threats to internal validity come mainly from experimental deviations due to parameter settings and the replication of the comparison methods. Most of the compared methods in the experiment are not available with their source codes, so some specific information like parameter settings is unknown and there is probably some variation in the baselines. Although we experimented exactly as mentioned in the literature, our reproducible results do not exactly match the original outcome. We overcame this threat by adjusting the parameter settings to choose the best combination of parameters.

The main external threat to validity is dataset bias. In this study, we selected 13 datasets from NASA MDP, Relink, and AEEEM for experimental validation of the proposed defect prediction model. However, it is still unknown how well the models that are constructed using this method will perform when applied to other applications.

6 Conclusions

To address the inefficiency of defect prediction models caused by the large variability of data distribution between the source project and target project, we propose a metric compensation method based on transfer learning and feature selection. First, peUpMeCom uses the Pearson feature selection method to filter out redundant features that are not relevant to the defect category prediction. The correlation coefficient between each feature and the defect category is calculated, and the irrelevant features are removed based on the correlation coefficient ranking to reduce training time of the model. After that, TCA is introduced to map the data features to a low-dimensional space, while preserving the original data characteristics. Then, using the bidirectional metric compensation technique, we enhance the feature distribution similarity between the source project and target project, thereby improving the performance of CPDP. Extensive experimental results verify that our peUpMeCom method effectively improves the prediction accuracy in both AUC value and F1-measure, and outperforms the traditional metric compensation techniques.

Contributors

Jinfu CHEN and Saihua CAI designed the research. Xiaoli WANG, Saihua CAI, and Jiaping XU processed the data. Jinfu CHEN, Xiaoli WANG, and Saihua CAI drafted the paper. Xiaoli WANG, Jiaping XU, Jingyi CHEN, and Haibo CHEN finished the experiments. Jingyi CHEN and Haibo CHEN helped organize the paper. Jinfu CHEN, Xiaoli WANG, and Saihua CAI revised and finalized the paper.

Compliance with ethics guidelines

Jinfu CHEN, Xiaoli WANG, Saihua CAI, Jiaping XU, Jingyi CHEN, and Haibo CHEN declare that they have no conflict of interest.

References

Amasaki S, Kawata K, Yokogawa T, 2015. Improving cross-project defect prediction methods with data simplification. Proc 41st Euromicro Conf on Software Engineering and Advanced Applications, p.96-103. https://doi.org/10.1109/SEAA.2015.25

Briand LC, Melo WL, Wüst J, 2002. Assessing the applicability of fault-proneness models across object-oriented software projects. *IEEE Trans Softw Eng*, 28(7):706-720. https://doi.org/10.1109/TSE.2002.1019484

Cai JC, Xu K, Zhu YH, et al., 2020. Prediction and analysis of net ecosystem carbon exchange based on gradient boosting regression and random forest. *Appl Energy*, 262:114566. https://doi.org/10.1016/j.apenergy.2020.114566

Chen JY, Yang YT, Hu KK, et al., 2019. Multiview transfer learning for software defect prediction. *IEEE Access*, 7:8901-8916. https://doi.org/10.1109/ACCESS.2018.2890733

Chen JY, Hu KK, Yu Y, et al., 2020. Software visualization and deep transfer learning for effective software defect prediction. Proc ACM/IEEE 42nd Int Conf on Software Engineering, p.578-589. https://doi.org/10.1145/3377811.3380389

Chen X, Zhao YQ, Wang QP, et al., 2018. MULTI: multi-objective effort-aware just-in-time software defect prediction. *Inform Softw Technol*, 93:1-13. https://doi.org/10.1016/j.infsof.2017.08.004

Fukushima T, Kamei Y, McIntosh S, et al., 2014. An empirical study of just-in-time defect prediction using cross-project models. Proc 11th Working Conf on Mining Software Repositories, p.172-181. https://doi.org/10.1145/2597073.2597075

Grimm LG, Nesselroade KPJr, 2018. Statistical Applications for the Behavioral and Social Sciences (2nd Ed.). John Wiley & Sons, Hoboken, USA.

Guo YC, Shepperd M, Li N, 2018. Bridging effort-aware prediction and strong classification: a just-in-time software defect prediction study. Proc 40th Int Conf on Software Engineering: Companion Proceeedings, p.325-326. https://doi.org/10.1145/3183440.3194992

Habibi PA, Amrizal V, Bahaweres RB, 2018. Cross-project defect prediction for web application using naive Bayes (case study: petstore web application). Proc Int Workshop on Big Data and Information Security, p.13-18. https://doi.org/10.1109/IWBIS.2018.8471701

Hall T, Beecham S, Bowes D, et al., 2012. A systematic literature review on fault prediction performance in software engineering. *IEEE Trans Softw Eng*, 38(6):1276-1304. https://doi.org/10.1109/TSE.2011.103

He P, Li B, Liu X, et al., 2015. An empirical study on software defect prediction with a simplified metric set. *Inform Softw Technol*, 59:170-190. https://doi.org/10.1016/j.infsof.2014.11.006

Herbold S, Trautsch A, Grabowski J, 2018. A comparative study to benchmark cross-project defect prediction approaches. Proc 40th Int Conf on Software Engineering, p.1063. https://doi.org/10.1145/3180155.3182542

Iqbal T, Cao Y, Kong QQ, et al., 2020. Learning with out-of-distribution data for audio classification. Proc IEEE Int Conf on Acoustics, Speech and Signal Processing, p.636-640. https://doi.org/10.1109/ICASSP40776.2020.9054444

Kamei Y, Fukushima T, McIntosh S, et al., 2016. Studying just-in-time defect prediction using cross-project models. *Empir Softw Eng*, 21(5):2072-2106. https://doi.org/10.1007/s10664-015-9400-x

Li K, Xiang ZL, Chen T, et al., 2020a. BILO-CPDP: bi-level programming for automated model discovery in cross-project defect prediction. Proc 35th IEEE/ACM Int Conf on Automated Software Engineering, p.573-584. https://doi.org/10.1145/3324884.3416617

Li K, Xiang ZL, Chen T, et al., 2020b. Understanding the automated parameter optimization on transfer learning

for cross-project defect prediction: an empirical study. Proc ACM/IEEE 42nd Int Conf on Software Engineering, p.566-577. https://doi.org/10.1145/3377811.3380360

Liu C, Yang D, Xia X, et al., 2019. A two-phase transfer learning model for cross-project defect prediction. *Inform Softw Technol*, 107:125-136. https://doi.org/10.1016/j.infsof.2018.11.005

Lv WD, 2019. Method and application of data defect analysis based on linear discriminant regression of far subspace. *Cluster Comput*, 22(2):4277-4282. https://doi.org/10.1007/s10586-018-1861-4

Madeyski L, Jureczko M, 2015. Which process metrics can significantly improve defect prediction models? An empirical study. *Softw Qual J*, 23(3):393-422. https://doi.org/10.1007/s11219-014-9241-7

Malhotra R, 2015. A systematic review of machine learning techniques for software fault prediction. *Appl Soft Comput*, 27:504-518. https://doi.org/10.1016/j.asoc.2014.11.023

Marian Z, Mircea IG, Czibula IG, et al., 2016. A novel approach for software defect prediction using fuzzy decision trees. Proc 18th Int Symp on Symbolic and Numeric Algorithms for Scientific Computing, p.240-247. https://doi.org/10.1109/SYNASC.2016.046

McBride R, Wang K, Ren ZY, et al., 2019. Cost-sensitive learning to rank. Proc 33rd AAAI Conf on Artificial Intelligence, p.4570-4577. https://doi.org/10.1609/aaai.v33i01.33014570

Nam J, Pan SJ, Kim S, 2013. Transfer defect learning. Proc 35th Int Conf on Software Engineering, p.382-391. https://doi.org/10.1109/ICSE.2013.6606584

Peng ML, Zhang Q, Xing XY, et al., 2019. Trainable undersampling for class-imbalance learning. Proc 33rd AAAI Conf on Artificial Intelligence, p.4707-4714. https://doi.org/10.1609/aaai.v33i01.33014707

Purnami SW, Trapsilasiwi RK, 2017. SMOTE-least square support vector machine for classification of multiclass imbalanced data. Proc 9th Int Conf on Machine Learning and Computing, p.107-111. https://doi.org/10.1145/3055635.3056581

Rahman F, Devanbu P, 2013. How, and why, process metrics are better. Proc 35th Int Conf on Software Engineering, p.432-441. https://doi.org/10.1109/ICSE.2013.6606589

Ryu D, Choi O, Baik J, 2014. Improving prediction robustness of VAB-SVM for cross-project defect prediction. Proc IEEE 17th Int Conf on Computational Science and Engineering, p.994-999. https://doi.org/10.1109/CSE.2014.198

Ryu D, Choi O, Baik J, 2016. Value-cognitive boosting with a support vector machine for cross-project defect prediction. *Empir Softw Eng*, 21(1):43-71. https://doi.org/10.1007/s10664-014-9346-4

Ryu D, Jang JI, Baik J, 2017. A transfer cost-sensitive boosting approach for cross-project defect prediction. *Softw Qual J*, 25(1):235-272. https://doi.org/10.1007/s11219-015-9287-1

Saidi R, Bouaguel W, Essoussi N, 2019. Hybrid feature selection method based on the genetic algorithm and Pearson correlation coefficient. In: Hassanien AE (Ed.), Machine Learning Paradigms: Theory and Application. Springer, Cham, p.3-24. https://doi.org/10.1007/978-3-030-02357-7_1

Shippey T, Bowes D, Hall T, 2019. Automatically identifying code features for software defect prediction: using AST *N*-grams. *Inform Softw Technol*, 106:142-160. https://doi.org/10.1016/j.infsof.2018.10.001

Shuai B, Li HF, Li MJ, et al., 2013. Software defect prediction using dynamic support vector machine. Proc 9th Int Conf on Computational Intelligence and Security, p.260-263. https://doi.org/10.1109/CIS.2013.61

Siers MJ, Islam Z, 2015. Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. *Inform Syst*, 51:62-71. https://doi.org/10.1016/j.is.2015.02.006

Tabassum S, Minku LL, Feng DY, et al., 2020. An investigation of cross-project learning in online just-in-time software defect prediction. Proc ACM/IEEE 42nd Int Conf on Software Engineering, p.554-565. https://doi.org/10.1145/3377811.3380403

Thejas GS, Garg R, Iyengar SS, et al., 2021. Metric and accuracy ranked feature inclusion: hybrids of filter and wrapper feature selection approaches. *IEEE Access*, 9:128687-128701. https://doi.org/10.1109/ACCESS.2021.3112169

Tsuda N, Washizaki H, Honda K, et al., 2019. WSQF: comprehensive software quality evaluation framework and benchmark based on SQuaRE. Proc IEEE/ACM 41st Int Conf on Software Engineering: Software Engineering in Practice, p.312-321. https://doi.org/10.1109/ICSE-SEIP.2019.00045

Wahono RS, 2015. A systematic literature review of software defect prediction: research trends, datasets, methods and frameworks. *J Softw Eng*, 1(1):1-16.

Wan ZY, Xia X, Hassan AE, et al., 2020. Perceptions, expectations, and challenges in defect prediction. *IEEE Trans Softw Eng*, 46(11):1241-1266. https://doi.org/10.1109/TSE.2018.2877678

Wang HJ, Khoshgoftaar TM, Napolitano A, 2010. A comparative study of ensemble feature selection techniques for software defect prediction. Proc 9th Int Conf on Machine Learning and Applications, p.135-140. https://doi.org/10.1109/ICMLA.2010.27

Watanabe S, Kaiya H, Kaijiri K, 2008. Adapting a fault prediction model to allow inter languagereuse. Proc 4th Int Workshop on Predictor Models in Software Engineering, p.19-24. https://doi.org/10.1145/1370788.1370794

Wu F, Jing XY, Dong XW, et al., 2017. Cross-project and within-project semi-supervised software defect prediction problems study using a unified solution. Proc IEEE/ACM 39th Int Conf on Software Engineering Companion, p.195-197. https://doi.org/10.1109/ICSE-C.2017.72

Yang XL, Lo D, Xia X, et al., 2017. TLEL: a two-layer ensemble learning approach for just-in-time defect prediction. *Inform Softw Technol*, 87:206-220. https://doi.org/10.1016/j.infsof.2017.03.007

Yu JL, Benesty J, Huang GP, et al., 2015. Optimal single-channel noise reduction filtering matrices from the Pearson correlation coefficient perspective. Proc IEEE Int Conf on Acoustics, Speech and Signal Processing, p.201-205. https://doi.org/10.1109/ICASSP.2015.7177960