2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)

MVSE: Effort-Aware Heterogeneous Defect Prediction via Multiple-View Spectral Embedding

Zhou Xu[1,2], Sizhe Ye[3], Tao Zhang[1*], Zhen Xia[3], Shuai Pang[3], Yong Wang[1], Yutian Tang[4]

[1]College of Computer Science and Technology, Harbin Engineering University, Harbin, China
[2]College of Informatics, Huazhong Agricultural University, Wuhan, China
[3]School of Computer Science, Wuhan University, Wuhan, China
[4]Department of Computing, The Hong Kong Polytechnic University, Hong Kong

Abstract—**Cross-Project Defect Prediction (CPDP) predicts defects in a target project using the defect information of the external project. Existing CPDP methods assume that the data of two projects share identical features. When cross-project data contain heterogeneous features, traditional CPDP methods become ineffective. In this paper, we propose a novel approach called Multiple-View Spectral Embedding (MVSE) to address the heterogeneous CPDP issue. MVSE treats the cross-project data as two different views and exploits the spectral embedding method to map the heterogeneous feature sets into a consistent space where the two mapped feature sets have maximal similarity. To evaluate MVSE in the realistic setting, we employ an effort-aware performance indicator that considers the cost of inspection in the context of heterogeneous CPDP scenario. We have conducted extensive experiments to compare MVSE with two state-of-the-art heterogeneous CPDP methods and within-project setting. The experiments on 94 cross-project pairs show that MVSE achieves promising results.**

Keywords: effort-aware evaluation; heterogeneous cross-project defect prediction; multiple-view learning; spectral embedding

I. Introduction

Software testing is the most expensive phase in software development life cycle [1]. Defect prediction can assist software quality assurance teams to identify the potential defects in the software entities (e.g. methods, classes or files) before releasing the software product. Effective defect prediction, which can improve software quality with limited test resources, has raised strong interests from the software engineering community over the past two decades [2].

The majority studies on defect prediction build prediction models on the labeled software entities, and then predict the defects in the unlabeled entities within the same project. This is referred as Within-Project Defect Prediction (WPDP). The labeled software entities consist of various features and defect information (e.g. a binary class label or the number of defects) that can be collected from historical project data in the software repositories such as version control systems (e.g. GitHub and Subversion) and issue tracking systems (e.g. JIRA and Bugzilla). However, WPDP is not applicable to new projects because there are very limited historical development data [3].

Recently, Cross-Project Defect Prediction (CPDP) has been proposed to address this issue. CPDP builds defect prediction model with the labeled data from other projects (i.e., source projects) to identify the defects in the unlabeled data of current projects (i.e., target project) [4]. Unfortunately, recent studies found that the overall results of CPDP are not satisfactory [3], [5] because cross-project data may not be similar and hence the models constructed on source project data will be unsuitable for target project data. Moreover, most previous studies on CPDP assume that the cross-project data share the same feature set. However, it is not always realistic for the source project data to have the same feature set as the target project data, because different projects may be developed with various programming languages and the features may be extracted with different granularity.

When cross-project data have heterogeneous feature sets, traditional CPDP methods are not effective. Recent studies try to alleviate this problem by exploiting the common feature set across projects [4], [7]. That is, only the common features are used for defect prediction while others are ignored. However, such method has three limitations. First, the discarded features may contain discriminative information for constructing classification models. Second, if the common feature set is small, there may not be enough useful information for accurate prediction [7]. Third, cross-project data may not have common features, such as completely heterogeneous CPDP [8], [9].

In this paper, we propose a Multiple-View Spectral Embedding (MVSE) approach for heterogeneous CPDP. MVSE is based on the observation that the cross-project data are related to the same object (i.e., software defect information), and thus we can treat the cross-project data as two diverse representations from different views towards this object.

Most previous studies have evaluated the efficacy of WPDP and CPDP with typical performance indicators, such as F-measure and AUC. These indicators do not consider the quality assurance effort required to review the predicted defective entities by assuming that there are enough resources to inspect all the entities [3], [4], [7], [10]. However, as suggest by Mende et al. [13], it is more realistic to use the effort-aware indicator for measuring the efficacy of defect prediction, because inspecting all the entities is often not feasible. Although effort-aware defect prediction has been considered before [11], [16], [18], we are the first to investigate effort-aware heterogeneous CPDP.

We have conducted extensive experiments to evaluate MVSE on 94 cross-project combinations of 12 projects from three public open-source datasets with an effort-aware indicator (i.e., **PofD**20). We also compare MVSE against two heterogeneous CPDP methods (i.e., CCA+ and HDP) and WPDP setting. The experimental results show that MVSE achieves average improvements of 32.3%, 32.3%, and 19.3% compared with the three methods, respectively.

[*] Corresponding author

Authorized licensed use limited to: Universidad Nacional de Costa Rica (UNA). Downloaded on June 07,2021 at 12:19:11 UTC from IEEE Xplore. Restrictions apply.

Our main contributions are highlighted as follows:

1) We propose a novel approach named MVSE to address heterogeneous CPDP with different features. MVSE regards this problem as a multiple-view learning task and employs spectral embedding to map the heterogeneous feature sets into a consistent and comparable feature space in which the two mapped feature sets have maximal similarity.
2) We propose using the effort-aware indicator to evaluate heterogeneous CPDP performance, which results in more realistic evaluation.
3) We conduct extensive experiments to evaluate MVSE by using 94 cross-project pairs and comparing it against two state-of-the-art heterogeneous CPDP methods and WPDP setting. The experimental results illustrate the superior of MVSE.

II. Related Work

A. Heterogenous Cross-Project Defect Prediction

Many CPDP methods have been proposed to address the challenges of defect prediction for projects that have no or limited historical unlabeled data [3], [7], [10]. When the source and target project data have completely different features, traditional CPDP methods cannot work since there are no common features to build prediction models. Here, we mainly introduce the two most typical studies related to this issue.

Jing et al. [8] proposed a method called CCA+ for heterogeneous CPDP. They applied canonical correlation analysis (CCA) technique to maximize the similarity of the heterogeneous target and source projects. However, CCA+ has several limitations. First, it needs to inspect whether the across project data have common features before applying CCA. If any, it selects them as common features. Second, if the dimensions of the features of the across-project data are different, CCA+ first uses the unified metric representative (UMR) technique to let the two data have the same number of features by unifying the feature representations. By contrast, MVSE neither needs to inspect whether there exist common features nor has a constraint on the dimension of the features. Moreover, Jing et al. [8] only reported the results of 32 cross-project pairs among total 90 cross-project combinations, thus the universality of CCA+ needs to be further verified.

Nam et al. [9] proposed the method called HDP for heterogeneous CPDP. HDP first uses typical feature selection techniques to eliminate useless features in the source project data, and then employs analyzers to match the selected features of the source project data with the features of the target project data, where the matched features have similar data distribution. A limitation of this method is that the potentially intrinsic structures hidden behind the data may not be sufficiently uncovered, because the matched features are not preprocessed with any data transformation. By contrast, MVSE maps the features into a common latent feature space where the mapped features between the cross-project data have maximal similarity.

B. Effort- Aware Defect Prediction

Although software quality assurance is critical, the resource, including time and man power, for this task is not unlimited. A balanced trade-off between cost and defect detection has to be achieved. In reality, we wish to inspect less code but find more defects. This effort is referred as effort-aware defect prediction. Typically, it is feasible to just inspect 5% codes under deadline pressure while 20% codes at other time periods [18].

Diverse features have been used to measure the inspection effort for the defects. For example, Arisholm et al. [11], Menzies et al. [12], Mende et al. [13], Jiang et al. [14] used the number of lines of code, while Ostrand et al. [37] employ the number of files. Kamei et al. [16] built an effort-aware just-in-time defect prediction model called EALR by treating the total number of lines of code modified by a change as the measurement of the effort to inspect the defects inducing changes. Experiments on 11 change-level projects showed that EALR can discover 35% changes by inspecting only 20% codes that are predicted to be defective. Rahman et al. [18] conducted WPDP and CPDP on 9 Apache projects and evaluated the performances with both effort-aware and traditional indicators. Experimental results showed that CPDP is significantly worse than WPDP in terms of the traditional indicators but outperforms WPDP in terms of the effort-aware indicator.

C. Spectral Embedding Approach

Spectral embedding based dimension reduction strives to extract a low-dimensional representation for high-dimensional data. This representation is derived from the top or bottom eigenvectors of specially constructed matrices [21]. Traditional dimension reduction methods usually assume that the data are represented with a single view. Spectral embedding methods, such as Principal Component Analysis [22], Isomap [23] and Locally Linear Embedding [24], can handle single-view data.

In our work, since each project data can be regarded as a single view, the heterogeneous cross-project data can be considered as multiple-view data that have different representations from multiple feature spaces. Multiple-view learning has been applied in many areas, such as multimedia domain [25] and word embedding [26]. Since traditional spectral methods cannot well handle multiple-view data, Long et al. [27] proposed DSE, a distributed spectral method to deal with multiple-view data. DSE embeds multiple-view data into a low-dimensional space where the single-view data are similar. Following this work, Shi et al. [28] proposed HeMap, a spectral method that maps the data into a common space where they have the maximal similarity. In MVSE, we adapt HeMap to the heterogeneous CPDP problem.

III. Method

In this section, we describe the details of the MVSE method. Let the target project data matrix be $\boldsymbol{T} = [t_1, t_2, \dots, t_m]'$, where $t_i = [t_{i1}, t_{i2}, \dots, t_{ip}] \in \mathbb{R}^p$. m and p denote the number of target project entities and that of features, respectively. T_{ij} indicates the jth feature of the ith target project entity. Similarly, let the source project data matrix be $\boldsymbol{S} = [s_1, s_2, \dots, s_n]'$, where $s_i = [s_{i1}, s_{i2}, \dots, s_{iq}] \in \mathbb{R}^q$, n and q denote the number of source project entities and that of features, respectively. S_{ij} indicates the jth feature of the ith source project entity. Due to the differences in the value ranges of diverse features, we use z-score technique [31] to normalize the data of the two projects.

Since in heterogeneous CPDP the target and source project data have different feature spaces (i.e., $\mathbb{R}^p \neq \mathbb{R}^q$), we map the target and source project data into a common latent feature space to maximize the similarity between the data of the two projects. For this purpose, we achieve the optimization objective in Eq(1) as follows:

$$G(\boldsymbol{B_T}, \boldsymbol{B_S}, \boldsymbol{T}, \boldsymbol{S}) = \min_{\boldsymbol{B_T}, \boldsymbol{B_S}} \ell(\boldsymbol{B_T}, \boldsymbol{T}) + \ell(\boldsymbol{B_S}, \boldsymbol{S}) + \beta \cdot \boldsymbol{D}(\boldsymbol{B_T}, \boldsymbol{B_S}), \quad (1)$$

Authorized licensed use limited to: Universidad Nacional de Costa Rica (UNA). Downloaded on June 07,2021 at 12:19:11 UTC from IEEE Xplore. Restrictions apply.

where $\boldsymbol{B_T} \in \mathbb{R}^{m\times k}$ and $\boldsymbol{B_s} \in \mathbb{R}^{n\times k}$ are the projected data matrices of target project data matrix $\boldsymbol{T}$ and source project data matrix $\boldsymbol{S}$, respectively. k indicates the number of projected features in the common space. $\ell(\boldsymbol{B_T}, \boldsymbol{T})$ (or $\ell(\boldsymbol{B_S}, \boldsymbol{S})$) is a distortion function to measure the difference between the mapped data matrix $\boldsymbol{B_T}$ (or $\boldsymbol{B_S}$) and original data matrix $\boldsymbol{T}$ (or $\boldsymbol{S}$). $\boldsymbol{D}(\boldsymbol{B_T}, \boldsymbol{B_S})$ is a function to measure the difference between the two mapped data matrices, and β is a parameter to control the difference degree.

The mapped data matrices $\boldsymbol{B_T}$ and $\boldsymbol{B_s}$ are deduced by performing linear transformations (such as rotation, scaling, and permutation operations) on the row vectors or column vectors of the original target project data matrix $\boldsymbol{T}$ and source project data matrix $\boldsymbol{S}$, respectively. Note that the order of the entities in $\boldsymbol{T}$ and $\boldsymbol{S}$ does not affect the results of the two mapped matrices [28]. The difference function between two mapped matrices $\boldsymbol{D}(\boldsymbol{B_T}, \boldsymbol{B_S})$ is defined as follows:

$$\boldsymbol{D}(\boldsymbol{B_T}, \boldsymbol{B_S}) = \frac{1}{2}(\ell(\boldsymbol{B_T}, \boldsymbol{S}) + \ell(\boldsymbol{B_S}, \boldsymbol{T})), \tag{2}$$

where $\ell(\boldsymbol{B_T}, \boldsymbol{S})$ denotes the difference between the mapped target project data matrix and the original source project data matrix, and $\ell(\boldsymbol{B_S}, \boldsymbol{T})$ denotes the difference between the mapped source project data matrix and the original target project data matrix. $\boldsymbol{D}(\boldsymbol{B_T}, \boldsymbol{B_S})$ is the mean of the two differences.

Eq (1) aims at maximizing the three function, i.e., the differences between$\boldsymbol{B_T}$ and $\boldsymbol{T}$, between $\boldsymbol{B_S}$ and $\boldsymbol{S}$, between $\boldsymbol{B_T}$ and $\boldsymbol{B_S}$. On one hand, the original structure information between the original project data and the corresponding mapped data will be preserved by minimizing the difference functions $\ell(\boldsymbol{B_T}, \boldsymbol{T})$ and $\ell(\boldsymbol{B_S}, \boldsymbol{S})$. On the other hand, the similarity of the two mapped project data $\boldsymbol{B_T}$ and $\boldsymbol{B_S}$ in the common latent feature space will be maximized by constraining the difference function $\boldsymbol{D}(\boldsymbol{B_T}, \boldsymbol{B_S})$. This optimization function strives to obtain the maximal similarity of the two mapped data without losing their original structure information. We solve this optimization problem using the spectral embedding method.

Spectral embedding aims at finding a latent mapped space using matrix transformation with linear operations. For the purpose of easily using a matrix form to express the difference between the two mapped data matrices, we need to preprocess the original target and source project data to let them have the same number of software entities (i.e., let $m = n$). This step can be easily done by two methods: (1) randomly increasing the number of entities in the smaller project dataset; or (2) decreasing the number of entities in the larger project dataset. Since the latter will lead to the information loss of the larger dataset due to eliminating some important entities, we choose the former. To preserve the distribution structure of the defective and non-defective entities, we proportionally grow the number of the two groups of entities in smaller project dataset. For example, if there is 50 entities difference between the initial two project data and the proportion of the defective and non-defective entities in the smaller dataset is 1:4, we will randomly sample 10 and 40 entities from the defective and non-defective entities of the smaller dataset, respectively, and add them into the smaller dataset.

Let $\boldsymbol{W_T} \in \mathbb{R}^{k\times p}$, $\boldsymbol{W_s} \in \mathbb{R}^{k\times q}$ denote the linear mapping matrices of the target project and source project data matrices, respectively. The difference functions $\ell(\boldsymbol{B_T}, \boldsymbol{T})$ and $\ell(\boldsymbol{B_S}, \boldsymbol{S})$ are defined as:

$$\ell(\boldsymbol{B_T}, \boldsymbol{T}) = ||\boldsymbol{B_T}\boldsymbol{W_T} - \boldsymbol{T}||^2, \tag{3}$$

$$\ell(\boldsymbol{B_S}, \boldsymbol{S}) = ||\boldsymbol{B_S}\boldsymbol{W_S} - \boldsymbol{S}||^2, \tag{4}$$

where $||\cdot||^2$ is the Frobenius norm.

With Eq (3) and Eq (4), Eq (1) can be rewritten as:

$$G(\boldsymbol{B_T}, \boldsymbol{B_S}, \boldsymbol{T}, \boldsymbol{S}) = \min_{\boldsymbol{B_T^T B_T = I, B_S^T B_S = I}} ||\boldsymbol{B_T W_T} - \boldsymbol{T}||^2 + ||\boldsymbol{B_S W_S} - \boldsymbol{S}||^2 + \frac{\beta}{2}(||\boldsymbol{B_S W_T} - \boldsymbol{T}||^2 + ||\boldsymbol{B_T W_S} - \boldsymbol{S}||^2), \tag{5}$$

Note that the linear transformations will be automatically performed on the matrix to minimize the corresponding differences [27]. The formulas of the optimal $\boldsymbol{W_T}$ and $\boldsymbol{W_S}$ in terms of $\boldsymbol{B_T}$ and $\boldsymbol{B_S}$ are deduced by:

$$\boldsymbol{W_T} = \frac{1}{2+\beta}(2 \cdot \boldsymbol{B_T'T} + \beta \cdot \boldsymbol{B_S'T}), \tag{6}$$

$$\boldsymbol{W_S} = \frac{1}{2+\beta}(2 \cdot \boldsymbol{B_S'S} + \beta \cdot \boldsymbol{B_T'S}), \tag{7}$$

where $\boldsymbol{B_T'}$ and $\boldsymbol{B_S'}$ are the transpose matrixes of $\boldsymbol{B_T}$ and $\boldsymbol{B_S}$.

With Eq (6) and Eq (7), Eq (5) can be rewritten as

$$G(\boldsymbol{B_T}, \boldsymbol{B_S}, \boldsymbol{T}, \boldsymbol{S}) = \min_{\boldsymbol{B_T^T B_T = I, B_S^T B_S = I}} \left(1 + \frac{\beta}{2}\right) tr(\boldsymbol{T'T}) + \left(1 + \frac{\beta}{2}\right) tr(\boldsymbol{S'S}) - \frac{1}{2+\beta} tr(\boldsymbol{B'AB}), \tag{8}$$

where $\mathbf{B} = \begin{bmatrix} \boldsymbol{B_T} \\ \boldsymbol{B_S} \end{bmatrix}$, $\mathbf{A} = \begin{bmatrix} \boldsymbol{A_1} & \boldsymbol{A_2} \\ \boldsymbol{A_3} & \boldsymbol{A_4} \end{bmatrix}$, $\boldsymbol{A_2} = \boldsymbol{A_3'} = \beta(\boldsymbol{SS'} + \boldsymbol{TT'})$, $\boldsymbol{A_1} = 2\boldsymbol{TT'} + \frac{\beta^2}{2}\boldsymbol{SS'}$, and $\boldsymbol{A_4} = 2\boldsymbol{SS'} + \frac{\beta^2}{2}\boldsymbol{TT'}$. $tr(\cdot)$ is the trace function to calculate the sum of the eigenvalues of a matrix. Since $tr(\boldsymbol{T'T})$ and $tr(\boldsymbol{S'S})$ are constants, the minimization of Eq (8) is equivalent to maximize the $tr(\boldsymbol{B'AB})$, i.e.,

$$G(\boldsymbol{B_T}, \boldsymbol{B_S}, \boldsymbol{T}, \boldsymbol{S}) = \max_{\boldsymbol{B'B=I}} tr(\boldsymbol{B'AB}), \tag{9}$$

Note that $\boldsymbol{TT'}$ and $\boldsymbol{SS'}$are symmetric matrices, so $\boldsymbol{A_1}$ and $\boldsymbol{A_4}$ are symmetric matrices. Then $\boldsymbol{A'}$=$\begin{bmatrix} \boldsymbol{A_1'} & \boldsymbol{A_3'} \\ \boldsymbol{A_2'} & \boldsymbol{A_4'} \end{bmatrix}$=$\begin{bmatrix} \boldsymbol{A_1} & \boldsymbol{A_2} \\ \boldsymbol{A_3} & \boldsymbol{A_4} \end{bmatrix} = \boldsymbol{A}$, so matrix $\boldsymbol{A}$ is also symmetric.

$\boldsymbol{B}$ is obtained by Ky-Fan theorem [29]. Assuming $\boldsymbol{A}$ as a symmetric matrix with l eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_l$ and the corresponding eigenvectors $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_l]$, this theorem proves that the optimal $\boldsymbol{B}$ of Eq (9) is given by the top k eigenvectors of $\boldsymbol{A}$, i.e., $\boldsymbol{B} = \boldsymbol{\alpha_k} = [\alpha_1, \ldots, \alpha_k]$, where k ($k \leq l$) is the desired number of mapped features in the common feature space. Hence, $\boldsymbol{B_T}$ is the first half rows of $\boldsymbol{\alpha_k}$, and $\boldsymbol{B_S}$ is the last half rows of $\boldsymbol{\alpha_k}$. Since the duplicate entities will be mapped into the same new entities, we eliminate the duplicate entities from the mapped data. Algorithm 1 summarizes the main steps of MVSE. Since the key step of MVSE is to resolve the eigenvectors of $\boldsymbol{A}$, its complexity is $O(l^3)$, where l is the dimension of $\boldsymbol{A}$. To facilitate researchers to reproduce our experiments, we release the replication package at *https://sites.google.com/view/mvse*.

Algorithm 1 Multiple-View Spectral Embedding (MVSE)

```
Input: target project data $\boldsymbol{T}$, source project data $\boldsymbol{S}$, controlled parameter $\beta$, dimensions of the selected features $k$
Output: the mapped target and source project data $\boldsymbol{B_T}$, $\boldsymbol{B_S}$
1. Use z-score technique to normalize $\boldsymbol{T}$ and $\boldsymbol{S}$.
2. Apply sampling technique to make the two data have the same number of entities.
3. Construct matrix $\boldsymbol{A}$.
4. Resolve the eigenvalues and eigenvectors $\boldsymbol{\alpha}$ of $\boldsymbol{A}$.
5. Obtain $\boldsymbol{B} = \boldsymbol{\alpha_k} = [\alpha_1, \ldots, \alpha_k]$ by selecting the top $k$ eigenvectors.
6. Return $\boldsymbol{B_T}$ as the first half rows of $\boldsymbol{B}$, $\boldsymbol{B_S}$ as the last half rows of $\boldsymbol{B}$.
7. Remove the duplicate entities in $\boldsymbol{B_T}$ and $\boldsymbol{B_S}$.
```

Authorized licensed use limited to: Universidad Nacional de Costa Rica (UNA). Downloaded on June 07,2021 at 12:19:11 UTC from IEEE Xplore. Restrictions apply.

IV. Experimental Setup

A. Benchmark Datasets

Our benchmark consists of 12 publicly available projects of three datasets including AEEEM [30], NetGene [32] and Eclipse [33]. Table I summarizes each project in the three datasets, including the feature granularity, the number of features (# Features), the number of entities (# Entities), the number of defective entities (# Defective), the percentage of defective entities (% Defective). For each dataset, we remove the non-numeric features and the features with only one value. In addition, we transfer the defect values into the binary features by labeling the software entity with at least one defect with value 1 and others with value 0. Although there is a common feature LOC (lines of code) between AEEEM and Eclipse, we still treat them as completely heterogeneous cross-project data.

Table I. The detailed description of each project in the three datasets

Datasets	Projects	Granularity	# Features	# Entities	# Defective	% Defective
NetGene	Httpclient	change	466	361	205	56.79%
	Jackrabbit		466	542	225	41.51%
	Lucene		466	1671	346	20.71%
	Rhino		466	253	109	43.08%
AEEEM	EQ	class	61	324	129	39.81%
	JDT		61	997	206	20.66%
	LC		61	691	64	9.26%
	ML		61	1862	245	13.16%
	PDE		61	1497	209	13.96%
Eclipse	p-2.0	package	210	377	190	50.40%
	p-2.1		210	434	194	44.70%
	p-3.0		210	661	313	47.35%

B. Performance Indicator

The effort-aware indicator has been recently proposed to evaluate different defect prediction models constrained by a limit effort [11], [16]. Such indicator is useful for the prediction setting because they consider the effort required to review the predicted defective entities [19].

In reality, we always want to maximize the benefit of any effort for quality assurance. In general, the effort is the number of lines of code that need to inspect, and the benefit is the number or percentage of defects discovered. In this work, we set the percentage of lines of code reviewed as 20% following prior studies [14], [17], [20]. Since this indicator measures the **P**ercent **of D**efects that can be detected when reviewing **20**% lines of code, we refer it as **PofD**20. Note that in other studies, this indicator is called **PofB**20 (**P**ercentage **of B**ugs) [14], [20] or **CE**20 (**C**ost **E**ffectiveness) [34], [35]. In order to calculate **PofD**20, we first rank the entities in a descending order based on the degree of their predicted risk values. The entities with high risk values deserve the high priority for inspecting. Then, we simulate a developer to review these entities one at a time in order, and count the percent of lines of code reviewed and the number of defects detected. We terminate the review process when 20% lines of code have been inspected, and calculate the percent of defects detected as **PofD**20. A higher **PofD**20 value indicates that more defects can be detected. In this work, we employ the definition of risk values in [13], [14], [20] (i.e., the probability outputs of a classification model for the entity) to calculate the effort-aware indicator.

C. Prediction Model.

We train a logistic regression model with the mapped source project data and then apply it to the mapped target project data. Logistic regression model is commonly used for defect prediction [15], [36], especially for effort-aware defect prediction [17], [34], [35].

D. Experiment Design

For AEEEM and Eclipse datasets, the LOC feature is treated as the measure of reviewing effort for quality assurance. NetGene dataset does not have the LOC feature. Note that the features in NetGene dataset are related to change transactions. Since previous studies used the total number of changed LOC [16], [17] or the number of files [37] to measure the effort, we employ the feature 'SumnumChangedFiles', which denotes the sum of changed files, to quantify the effort for NetGene dataset.

For the dimensions of the selected features k, we do not specify a constant value for all the cross-project cases. For different cross-project pairs, we empirically maintain the top eigenvectors whose sum of corresponding eigenvalues occupies at least 50% of total sum of all the eigenvalues. For example, if k is the minimal value that satisfies the inequality $\alpha_1 + \alpha_1 +, \dots, +\alpha_k \geq 0.5 * (\alpha_1 + \alpha_1 +, \dots, +\alpha_l)$, then k is the selected dimensions of mapped target and source project data. The threshold 50% can be specified based on different requests. For the controlled parameter β, due to lacking of prior knowledge, we set it as 1 which is not partial to any objective function.

Due to the random sampling in MVSE, for each cross-project pair, we repeat MVSE 30 times to alleviate the potential bias of sampling, which may affect the experimental results. Then, we report the means of the effort-aware indicator.

V. Results

A. RQ1: How does MVSE perform in effort-aware heterogeneous CPDP?

Since projects in the same dataset have homogeneous features, to perform heterogeneous CPDP, for each target project selected from one dataset, each project in the other two datasets will be selected as the source project. Therefore, we obtain 94 (4*(5+3) + 5*(4+3) + 3(4+5) = 94) cross-project pairs. To evaluate the effectiveness of MVSE, we compare MVSE with two state-of-the-art baseline methods (CCA+ in [8] and HDP in [9]) and the typical WPDP setting. Since CCA+ and HDP methods do not involve sampling, we run the two methods only once. In terms of CCA+, as the authors did not mention the number of features that they selected in the mapped feature space, for a fair comparison, we also use the threshold 50% to select the number of mapped features. For HDP, the authors applied four feature selection methods in the feature selection phase and three analyzers and ten cutoff thresholds in the feature matching phase. We choose significance attribute selection method for feature selection stage and KSAnalyzer with the cutoff threshold of 0.05 for feature matching stage, which has led to the best performance for heterogeneous CPDP among all combinations. For WPDP, we divide the entities in a project into two sets: one used to train a prediction model and the other used to evaluate the performance of the model. We randomly select 50% of entities in the target project data as the training data and the re-

Authorized licensed use limited to: Universidad Nacional de Costa Rica (UNA). Downloaded on June 07,2021 at 12:19:11 UTC from IEEE Xplore. Restrictions apply.

maining entities as the test data. This setting follows the previous studies [8], [38], [39]. We repeat the process 30 times to eliminate the bias of the randomness of sampling and report the average indicator values. For a fair comparison, we also conduct WPDP with the logistic regression model as used in MVSE.

Table II. The **PofD**20 values of MVSE and three baseline methods when the projects in NetGene dataset as the target projects

Source	Target	MVSE	CCA+	HDP	WPDP
EQ	Httpclient	**0.387** ± 0.002	0.298	0.22	0.282
JDT		**0.378** ± 0.004	0.18	0.263	
LC		**0.381** ± 0.003	0.224	0.288	
ML		**0.379** ± 0.005	0.185	0.312	
PDE		**0.381** ± 0.004	0.224	0.356	
p-2.0		**0.356** ± 0.003	0.293	0.341	
p-2.1		**0.348** ± 0.003	0.288	0.215	
p-3.0		0.358 ± 0.005	0.288	**0.4**	
EQ	Jackrabbit	**0.476** ± 0.001	0.44	0.28	0.364
JDT		**0.481** ± 0.010	0.347	0.449	
LC		**0.477** ± 0.006	0.236	0.369	
ML		**0.471** ± 0.004	0.222	0.427	
PDE		**0.482** ± 0.011	0.316	0.396	
p-2.0		0.470 ± 0.002	0.329	**0.48**	
p-2.1		**0.476** ± 0.000	0.4	0.36	
p-3.0		**0.445** ± 0.005	0.382	0.378	
EQ	Lucene	0.389 ± 0.001	**0.457**	0.168	0.319
JDT		0.505 ± 0.001	**0.983**	0.367	
LC		0.425 ± 0.001	**0.983**	0.454	
ML		0.418 ± 0.011	**0.558**	0.454	
PDE		0.437 ± 0.001	**0.564**	0.402	
p-2.0		**0.435** ± 0.001	0.413	0.387	
p-2.1		**0.452** ± 0.004	0.434	0.35	
p-3.0		**0.425** ± 0.003	0.335	0.327	
EQ	Rhino	0.409 ± 0.024	0.349	**0.422**	0.303
JDT		**0.383** ± 0.010	0.138	0.193	
LC		**0.387** ± 0.010	0.119	0.211	
ML		**0.385** ± 0.008	0.165	0.284	
PDE		**0.385** ± 0.008	0.147	0.193	
p-2.0		**0.384** ± 0.007	0.376	0.266	
p-2.1		**0.385** ± 0.002	0.312	0.156	
p-3.0		**0.398** ± 0.041	0.312	0.257	
Average		**0.417**	0.353	0.326	0.317
p-value		--	3.13E-03	8.00E-06	--
δ value		--	0.535	0.584	--

To statistically analyze the results from MVSE and the baseline methods (mainly CCA+ and HDA), we perform the Wilcoxon test [40] to check whether the performance values of MVSE are significantly different from others at a confidence level 95%. The difference is statistically significant if the *p* value of the Wilcoxon test is less than 0.05. In addition, we follow the work in [41], [42] to calculate the Cliff's Delta (δ) effect size to quantify the amount of the difference. The difference is substantial if the δ value is equal to or greater than 0.474 [43].

Table II, III and IV report the **PofD**20 values of MVSE, CCA+, HDP, and MVSE in three heterogeneous prediction scenarios where the projects in NetGene, AEEEM and Eclipse are used as the target projects, respectively. We also report the standard deviations of **PofD**20 by MVSE, and the statistic test results. The best **PofD**20 values are in bold. From the tables, we have the following findings:

Table III. The **PofD**20 values of MVSE and the three baseline methods when the projects in AEEEM dataset as the target projects

Source	Target	MVSE	CCA+	HDP	WPDP
Httpclient	EQ	**0.496** ± 0.000	0.186	0.225	0.355
Jackrabbit		**0.504** ± 0.000	0.186	0.295	
Lucene		**0.496** ± 0.000	0.186	0.341	
Rhino		0.499 ± 0.007	**0.512**	0.341	
p-2.0		**0.507** ± 0.005	0.186	0.349	
p-2.1		**0.493** ± 0.004	0.186	0.357	
p-3.0		0.507 ± 0.004	**0.512**	0.357	
Httpclient	JDT	0.361 ± 0.009	0.359	0.248	**0.450**
Jackrabbit		0.353 ± 0.008	0.238	0.233	
Lucene		0.364 ± 0.000	0.238	0.238	
Rhino		0.356 ± 0.010	0.238	0.33	
p-2.0		0.363 ± 0.003	0.238	0.325	
p-2.1		0.361 ± 0.003	0.238	0.301	
p-3.0		0.352 ± 0.002	0.238	0.335	
Httpclient	LC	0.491 ± 0.017	**0.516**	0.344	0.389
Jackrabbit		**0.477** ± 0.012	0.313	0.156	
Lucene		**0.469** ± 0.000	0.313	**0.469**	
Rhino		0.506 ± 0.021	**0.516**	0.438	
p-2.0		**0.501** ± 0.008	0.313	0.297	
p-2.1		**0.496** ± 0.007	0.313	0.344	
p-3.0		**0.516** ± 0.003	**0.516**	0.344	
Httpclient	ML	0.404 ± 0.009	0.392	0.261	**0.485**
Jackrabbit		0.398 ± 0.007	0.237	0.269	
Lucene		0.398 ± 0.004	0.237	0.286	
Rhino		0.405 ± 0.012	0.392	0.241	
p-2.0		0.406 ± 0.007	0.237	0.273	
p-2.1		0.389 ± 0.005	0.237	0.314	
p-3.0		0.408 ± 0.005	0.392	0.339	
Httpclient	PDE	0.292 ± 0.010	0.234	0.254	**0.320**
Jackrabbit		0.309 ± 0.010	0.234	0.182	
Lucene		0.282 ± 0.000	0.234	0.268	
Rhino		0.308 ± 0.012	0.282	0.287	
p-2.0		0.300 ± 0.008	0.234	0.273	
p-2.1		0.300 ± 0.005	0.234	0.273	
p-3.0		0.284 ± 0.002	0.282	0.278	
Average		**0.410**	0.297	0.299	0.400
p-value		--	3.00E-06	3.65E-07	--
δ value		--	0.583	0.748	--

Authorized licensed use limited to: Universidad Nacional de Costa Rica (UNA). Downloaded on June 07,2021 at 12:19:11 UTC from IEEE Xplore. Restrictions apply.

First, in table II, the **PofD**20 values by MVSE vary from 0.348 to 0.505 when the projects in NetGene dataset act as target projects. Across the 32 cross-project pairs, the average **PofD**20 value by MVSE (0.417) makes an improvement by 18.1%, 27.9%, and 31.5% compared with CCA+, HDP, and WPDP, respectively.

Table IV. The **PofD**20 values of MVSE and the three baseline methods when the projects in Eclipse dataset as the target projects

Source	Target	MVSE	CCA+	HDP	WPDP
EQ	p-2.0	**0.447** ± 0.000	0.426	0.337	0.348
JDT		**0.427** ± 0.009	0.221	0.326	
LC		**0.403** ± 0.019	0.247	0.263	
ML		**0.454** ± 0.008	0.247	0.405	
PDE		**0.444** ± 0.010	0.184	0.311	
Httpclient		**0.437** ± 0.002	0.247	0.342	
Jackrabbit		**0.411** ± 0.007	0.247	0.384	
Lucene		0.429 ± 0.004	**0.463**	0.442	
Rhino		**0.422** ± 0.009	0.253	0.284	
EQ	p-2.1	**0.496** ± 0.002	0.356	0.263	0.337
JDT		**0.459** ± 0.008	0.356	0.376	
LC		**0.399** ± 0.015	0.356	0.299	
ML		**0.460** ± 0.009	0.356	0.438	
PDE		**0.466** ± 0.011	0.356	0.325	
Httpclient		0.477 ± 0.006	**0.479**	0.34	
Jackrabbit		0.469 ± 0.008	**0.479**	0.351	
Lucene		**0.470** ± 0.005	0.356	0.438	
Rhino		**0.488** ± 0.008	0.247	0.356	
EQ	p-3.0	**0.500** ± 0.002	0.351	0.256	0.355
JDT		**0.465** ± 0.005	0.233	0.364	
LC		**0.463** ± 0.002	0.211	0.294	
ML		**0.472** ± 0.010	0.217	0.444	
PDE		**0.468** ± 0.008	0.233	0.358	
Httpclient		0.489 ± 0.005	**0.495**	0.31	
Jackrabbit		**0.475** ± 0.004	0.233	0.361	
Lucene		**0.466** ± 0.004	0.233	0.425	
Rhino		0.466 ± 0.007	**0.495**	0.288	
Average		**0.456**	0.318	0.347	0.347
p-value		--	3.60E-05	6.00E-06	--
δ value		--	0.691	0.905	--

Second, in table III, the **PofD**20 values by MVSE vary from 0.282 to 0.516 when the projects in AEEEM dataset serve as the target projects. In terms of average **PofD**20 across the 35 cross-project pairs, MVSE (0.410) obtains improvements by 38%, 37.1%, and 2.5% compared with CCA+, HDP, and WPDP, respectively

Third, in table IV, the **PofD**20 values by MVSE vary from 0.399 to 0.500 when the projects in Eclipse dataset behave as target projects. In terms of the average **PofD**20 across the 27 cross-project pairs, MVSE (0.456) outperform CCA+, HDP, and WPDP by 43.4%, 31.4%, and 31.4%, respectively.

Table V. The average correlation coefficient of the selected feature pairs between each cross-project data

Cross Project Pair		Pearson	Cross Project Pair		Pearson
EQ	Httpclient	0.993 ± 0.001	EQ	Rhino	0.997 ± 0.001
JDT	Httpclient	1.000 ± 0.000	JDT	Rhino	1.000 ± 0.000
LC	Httpclient	1.000 ± 0.000	LC	Rhino	1.000 ± 0.000
ML	Httpclient	1.000 ± 0.000	ML	Rhino	1.000 ± 0.000
PDE	Httpclient	1.000 ± 0.000	PDE	Rhino	1.000 ± 0.000
p-2.0	Httpclient	0.986 ± 0.001	p-2.0	Rhino	0.899 ± 0.008
p-2.1	Httpclient	0.940 ± 0.014	p-2.1	Rhino	0.897 ± 0.007
p-3.0	Httpclient	0.986 ± 0.010	p-3.0	Rhino	0.950 ± 0.017
EQ	Jackrabbit	0.997 ± 0.001	p-2.0	EQ	0.997 ± 0.001
JDT	Jackrabbit	1.000 ± 0.000	p-2.1	EQ	0.989 ± 0.001
LC	Jackrabbit	1.000 ± 0.000	p-3.0	EQ	0.993 ± 0.003
ML	Jackrabbit	1.000 ± 0.000	p-2.0	JDT	0.997 ± 0.002
PDE	Jackrabbit	1.000 ± 0.000	p-2.1	JDT	0.992 ± 0.006
p-2.0	Jackrabbit	0.992 ± 0.004	p-3.0	JDT	0.989 ± 0.005
p-2.1	Jackrabbit	0.996 ± 0.006	p-2.0	LC	0.989 ± 0.005
p-3.0	Jackrabbit	0.992 ± 0.007	p-2.1	LC	0.991 ± 0.007
EQ	Lucene	0.999 ± 0.001	p-3.0	LC	0.990 ± 0.001
JDT	Lucene	1.000 ± 0.000	p-2.0	ML	0.998 ± 0.002
LC	Lucene	1.000 ± 0.000	p-2.1	ML	0.994 ± 0.006
ML	Lucene	1.000 ± 0.000	p-3.0	ML	0.995 ± 0.002
PDE	Lucene	1.000 ± 0.000	p-2.0	PDE	0.995 ± 0.003
p-2.0	Lucene	0.994 ± 0.005	p-2.1	PDE	0.995 ± 0.003
p-2.1	Lucene	0.995 ± 0.003	p-3.0	PDE	0.996 ± 0.004
p-3.0	Lucene	0.989 ± 0.006			

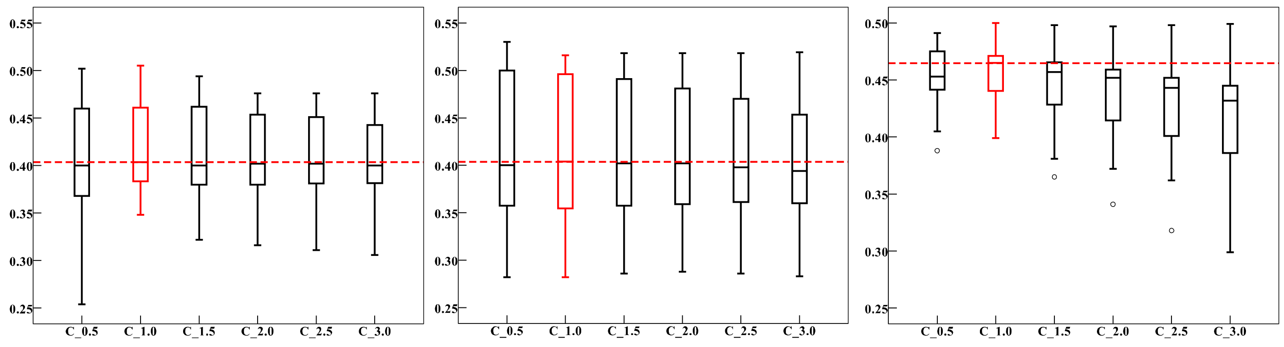
Fourth, from the three tables, we find that when the projects in Eclipse dataset are used as the target projects, the **PofD**20 values of all the cross-project pairs keep in the range from 0.4 to 0.5 (except for the cross-project pair between LC and p-2.1), which are more stable than that when the projects of other two datasets are used as the target projects. Moreover, the average **PofD**20 value is the highest among the three scenarios.

Fifth, when Httpclient, Rhino, JDT and PDE are used as the target projects, no matter which project is used as the source project, the **PofD**20 values are no more than 0.4 with only one exception for the cross-project pair between EQ and Rhino. It implies that we cannot find appropriate heterogeneous source project among the studied datasets to get relatively higher **PofD**20 values when the four projects are the target projects.

Sixth, in terms of CCA+ and HDP, for a target project with different source projects, the **PofD**20 values have relatively large differences in many cases. For example, in terms of CCA+ method, when Lucene project serve as the target project, the **PofD**20 values are 0.983 and 0.335 when the JDT and p-3.0 are the source projects, respectively; in terms of HDP method, when LC project act as the target project, the **PofD**20 values are 0.469 and 0.156 when the Lucene and Jackrabbit are the source projects, respectively. It signifies that CCA+ and HDP achieve unstable performances compared with MVSE in terms of **PofD**20. In terms of WPDP, when all projects in NetGene dataset, Eclipse dataset, and EQ and LC in AEEEM dataset act as the target projects, in terms of **PofD**20, MVSE shows overwhelming superiority compared with WPDP, but when other projects in AEEEM dataset are the target projects, WPDP is better on all the cross-project pairs.

Seventh, the p values and δ values indicate that the differences of **PofD**20 values among MVSE, CCA+ and HDP are statistically significant and substantial.

Authorized licensed use limited to: Universidad Nacional de Costa Rica (UNA). Downloaded on June 07,2021 at 12:19:11 UTC from IEEE Xplore. Restrictions apply.



(a) NetGene dataset (b) AEEEM dataset (c) Eclipse dataset

Figure 1. The boxplots of **PofD** 20 by MVSE under 6 controlled parameter β.

To sum up, the **PofD**20 values achieved by MVSE are superior to that by 3 baseline methods on most cross-project pairs, especially when the projects in NetGene and Eclipse act as the target projects.

B. RQ2: Why MVSE Works Well for Heterogeneous CPDP?

As mentioned above, MVSE maps the target and source project data into a common feature space where the mapped cross-project data have the maximal similarity. To measure the degree of the similarity, we use the Pearson correlation coefficient [6] to calculate the correlations of the feature pairs between the mapped target and source project data. For example, let the mapped target and source project data have k features with N entities, and let f_t and f_s be the jth ($j \in \{1,2,\dots,k\}$) feature of the mapped target and source project data. The Pearson correlation coefficient $r_{f_t f_s}$ is defined as

$$r_{f_t f_s} = \frac{\sum_{i=1}^{N}(f_{ti}-\overline{f_t})(f_{si}-\overline{f_s})}{\sqrt{\sum_{i=1}^{N}(f_{ti}-\overline{f_t})^2 \cdot \sum_{i=1}^{N}(f_{si}-\overline{f_s})^2}}, \quad (10)$$

where $\overline{f_t} = \frac{1}{N}\sum_{i=1}^{N} f_{ti}$ and $\overline{f_s} = \frac{1}{N}\sum_{i=1}^{N} f_{si}$. The greater coefficient value means stronger degree of correlation.

Table V lists the average correlation coefficients of the feature pairs between each cross-project data. We find that:

First, all the average coefficient values are great than 0.89, showing great similarity among the feature pairs between cross-project data. In most cases, the feature pairs have powerful similarity with average coefficient values more than 0.98.

Second, when the cross-project pairs are based on the projects from AEEEM and NetGene, the average coefficient values are very close to 1. It implies that MVSE can establish very strong correlations between the projects of the two datasets.

These observations indicate that MVSE can learn a better representative for the heterogeneous cross-project data with strong similarity. Thus, the mapped source project data are conductive to build an effective prediction model for the mapped target project data.

C. How various controlled parameter β impact the efficacy of MVSE for the effort-aware heterogeneous CPDP?

Due to lacking of prior knowledge towards how we desire the degree of the difference of two mapped data, we set the controlled parameter β as 1 as mentioned in Section IV-D. Lower parameter values mean that we expect the two mapped data are more similar. Here, we explore the performances of MVSE with different β values. For the purpose, we empirically select six different controlled parameters: 0.5 (C_0.5), 1.0 (C_1.0), 1.5 (C_1.5), 2.0 (C_2.0), 2.5 (C_2.5) and 3.0 (C_3.0).

Figure 1 only shows the boxplots of **PofD**20 on three heterogeneous cross-project scenarios with the six controlled parameter values. From the figure, we find that, when the projects in NetGene and AEEEM datasets sever as the target projects, the median **PofD**20 values by MVSE with β=1 are only a little higher than that with other 5 parameters, and when the projects in Eclipse dataset are the target projects, the median **PofD**20 values by MVSE with β=1 are obvious superior to that with other 5 parameters. Hence, β=1 is the optimum option to enable MVSE to achieve better performances.

VI. THREAT TO VALIDITY

We evaluate our method MVSE on 12 projects from three public open-source software defect datasets. Thus, the external threat is that the results of our experiments may not be generalized to other datasets. However, we carefully select the datasets with heterogeneous features at different levels of granularity to make our conclusions more general. Experiments on additional defect data are needed in future. For the baseline methods, since the authors do not provide the source codes, we carefully implement CCA+ and HDP following the descriptions of the methods in the original papers to minimize the potential inconsistency.

VII. CONCLUSION

Heterogeneous CPDP is a challenging issue due to inconsistent features between the cross-project data for building prediction models, and hence traditional CPDP methods cannot be directly applied to it. In this work, we treat heterogeneous CPDP as a multiple-view learning task and propose a novel method named MVSE to address it. MVSE maps the cross-project data into a space with spectral embedding method to make the heterogeneous features consistent and comparable. We conduct extensive experiments on 94 heterogeneous cross-project pairs to evaluate MVSE with an effort-aware indicator. Experimental results show that MVSE performs well in terms of **PofD**20 indicator compared with WPDP and two state-of-the-art heterogeneous CPDP methods in most cases.

In future work, we will evaluate MVSE on more heterogeneous datasets and with more effort-aware performance indicators. In addition, we will incorporate the class imbalance issue into our heterogeneous CPDP method.

Authorized licensed use limited to: Universidad Nacional de Costa Rica (UNA). Downloaded on June 07,2021 at 12:19:11 UTC from IEEE Xplore. Restrictions apply.

References

[1] M. E. Khan and F. Khan. Importance of Software Testing in Software Development Life Cycle. *International Journal of Computer Science Issues*, 2014.

[2] 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*, 38(6): 1276-1304, 2012.

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

[4] Y. Ma, G. Luo, X. Zeng, and A. Chen. Transfer learning for cross-company software defect prediction. *Information and Software Technology (IST)*, 54(3): 248-256, 2012.

[5] L. C. Briand, W. L. Melo, and J. Wust. Assessing the applicability of fault-proneness models across object-oriented software projects. *IEEE Transactions on Software Engineering*, 28(7): 706-720, 2002.

[6] J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. *Noise reduction in speech processing*. Springer Berlin Heidelberg, 1-4, 2009.

[7] B. Turhan, T. Menzies, A. B. Bener, and J. D. Stefano. On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, 14(5): 540-578, 2009.

[8] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu. Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning. *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (FSE)*. ACM, 496-507, 2015.

[9] J.Nam and S. Kim. Heterogeneous defect prediction. *Proceedings of the 10th joint meeting on Foundations of Software Engineering (FSE)*. ACM, 508-519, 2015.

[10] L. Chen, B. Fang, Z. Shang, and Y. Tang. Negative samples reduction in cross-company software defects prediction. *Information and Software Technology*, 62: 67-77, 2015.

[11] E. Arisholm, L. C. Briand, and E. B. ohannessen. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. *Journal of Systems and Software*, 83(1): 2-17, 2010.

[12] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener. Defect prediction from static code features: current results, limitations, new approaches. *Automated Software Engineering*, 17(4): 375-407, 2010.

[13] T. Mende and R. Koschke. Effort-aware defect prediction models. *Proceedings of the 14th European Conference on Software Maintenance and Reengineering (CSMR)*. IEEE, 2010: 107-116.

[14] T. Jiang, L. Tan, and S. Kim. Personalized defect prediction. *Proceedings of the 28th International Conference on Automated Software Engineering (ASE)*. IEEE, 279-289, 2013.

[15] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, 34(4): 485-496, 2008.

[16] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi. A large-scale empirical study of just-in-time quality assurance. *IEEE Transactions on Software Engineering*, 757-773, 2013.

[17] X. Yang, D. Lo, X. Xia, Y. Zhang, and J. Sun. Deep learning for just-in-time defect prediction. *International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 17-26, 2015.

[18] F. Rahman, D. Posnett, and P. Devanbu. Recalling the imprecision of cross-project defect prediction. *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE)*. ACM, 61, 2012.

[19] Y. Yang, Y. Zhou, J. Liu, Y. Zhao, H. Lu, L. Xu, B. Xu, and H. Leung. Effort-aware just-in-time defect prediction: simple unsupervised models could be better than supervised models. *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. ACM, 157-168, 2016.

[20] X. Xia, D. Lo, S. J. Pan, N. Nagappan, and X. Wang. HYDRA: Massively compositional model for cross-project defect prediction. *IEEE Transactions on Software Engineering*, 42(10): 977-998, 2016.

[21] L. K. Saul, K. Q. Weinberger, J. H. Ham, and F. Sha. Spectral methods for dimensionality reduction. *Semisupervised Learning*, 293-308, 2006.

[22] I. Jolliffe. Principal component analysis. John Wiley & Sons, Ltd, 2002.

[23] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500): 2319-2323, 2000.

[24] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500): 2323-2326, 2000.

[25] B. Eilam and Y. Poyas. Learning with multiple representations: Extending multimedia learning beyond the lab. *Learning and instruction*, 18(4): 368-378, 2008.

[26] P. Dhillon, D. P. Foster, and L. H. Ungar. Multi-view learning of word embeddings via cca. *Advances in Neural Information Processing Systems*. 199-207, 2011.

[27] B. Long, P. S. Yu, and Z. Zhang. A general model for multiple view unsupervised learning. Proceedings of the 2008 SIAM international conference on data mining. *Society for Industrial and Applied Mathematics*, 822-833, 2008.

[28] X. Shi, Q. Liu, W. Fan, and P. S. Yu. Transfer across completely different feature spaces via spectral embedding. *IEEE Transactions on Knowledge and Data Engineering*, 25(4): 906-918, 2013.

[29] R. Bhatia. Matrix analysis. Springeer-Cerlag, New York, 1997.

[30] M. D'Ambros, M. Lanza, and R. Robbes. An extensive comparison of bug prediction approaches. *International Working Conference on Mining Software Repositories (MSR)*, 31-41, 2010.

[31] Kotsiantis S B, Kanellopoulos D, and Pintelas P E. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2): 111-117, 2006.

[32] K. Herzig, S. Just, A. Rau, and A. Zeller. Predicting defects using change genealogies. *Proceedings of the 24th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 118-127, 2013.

[33] T. Zimmermann, R. Premraj, and A. Zeller. Predicting defects for eclipse. Proceedings of the third international workshop on predictor models in software engineering. *IEEE Computer Society*, 9, 2007.

[34] Y. Yang, Y. Zhou, H. Lu, L. Chen, Z. Chen, B. Xu, H. Leung, and Z. Zhang. Are slice-based cohesion metrics actually useful in effort-aware post-release fault-proneness prediction? An empirical study. *IEEE Transactions on Software Engineering*, 41(4): 331-357, 2015.

[35] Y. Yang, M. Harman, J. Krinke, S. Islam, D. Binkley, Y. Zhou, and B. Xu. An empirical study on dependence clusters for effort-aware fault-proneness prediction. *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 296-307, 2016.

[36] A. Mockus, P. Zhang, P. L. Li. Predictors of customer perceived software quality. *Proceedings of the 27th International Conference on Software Engineering (ICSE)*. ACM, 225-233, 2005.

[37] T. J. Ostrand, E. J. Weyuker, and R. M. Bell. Predicting the location and number of faults in large software systems. *IEEE Transactions on Software Engineering*, 31(4): 340-355, 2005.

[38] T. Wang, Z. Zhang, X. Jing, and L. Zhang. Multiple kernel ensemble learning for software defect prediction. *Automated Software Engineering*, 23(4): 569-590, 2016.

[39] X. Y. Jing, S. Ying, Z. W. Zhang, S. S. Wu, and J. Liu. Dictionary learning based software defect prediction. *Proceedings of the 36th International Conference on Software Engineering (ICSE)*. ACM, 414-423, 2014.

[40] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 80-83, 1945.

[41] P. He, B. Li, X. Liu, J. Chen, and Y. Ma. An empirical study on software defect prediction with a simplified metric set. *Information and Software Technology*, 59: 170-190, 2015.

[42] P. He, B. Li, and Y. Ma. Towards cross-project defect prediction with imbalanced feature sets. *CoRR, abs/* 1411.4228, 2014.

[43] N. Cliff. Ordinal methods for behavioral data analysis. *Psychology Press*, 2014.

Authorized licensed use limited to: Universidad Nacional de Costa Rica (UNA). Downloaded on June 07,2021 at 12:19:11 UTC from IEEE Xplore. Restrictions apply.