IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 25, 2018, accepted September 28, 2018, date of publication October 4, 2018, date of current version October 29, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2873755

HDA: Cross-Project Defect Prediction via Heterogeneous Domain Adaptation With Dictionary Learning

ZHOU XU[1,2,3], PEIPEI YUAN[4], TAO ZHANG[1,5], YUTIAN TANG[3], SHUAI LI[3], AND ZHEN XIA[2]

[1]College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
[2]School of Computer Science, Wuhan University, Wuhan 430072, China
[3]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[4]School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China
[5]Key Laboratory of Network Assessment Technology, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100190, China

Corresponding author: Tao Zhang (cstzhang@hrbeu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602258, in part by the China Postdoctoral Science Foundation under Grant 2017M621247, in part by the Heilongjiang Postdoctoral Science Foundation under Grant LBH-Z17047, and in part by the Fundamental Research Funds for the Central Universities under Grant HEUCFJ170604.

ABSTRACT Cross-Project Defect Prediction (CPDP) is an active topic for predicting defects on projects (target projects) with scarce-labeled data by reusing the classification models from other projects (source projects). Traditional CPDP methods require common features between the data of two projects and utilize them to construct defect prediction models. However, when cross-project data do not satisfy the requirement, i.e., heterogeneous CPDP (HCPDP) scenario, these methods become infeasible. In this paper, we propose a novel HCPDP method called Heterogeneous Domain Adaptation (HDA) to address the issue. HDA treats the cross-project data as being from two different domains with heterogeneous feature sets. It employs the domain adaptation method to embed the data from the two domains into a comparable feature space with a lower dimension, then measures the difference between the two mapped domains of data using the dictionaries learned from them with the dictionary learning technique. We comprehensively evaluate HDA on 94 cross-project pairs of 12 projects from three open-source defect data sets with three performance indicators, i.e., F-measure, Balance, and AUC. Compared with the two state-of-the-art HCPDP methods, the experimental results indicate that HDA improves 0.219 and 0.336 in terms of F-measure, 0.185 and 0.215 in terms of Balance, and 0.131 and 0.035 in terms of AUC. In addition, HDA achieves comparable results compared with Within-Project Defect Prediction (WPDP) setting and a state-of-the-art unsupervised learning method in most cases.

INDEX TERMS Heterogeneous cross-project defect prediction, heterogeneous domain adaptation, dictionary learning.

I. INTRODUCTION

With the continued growth of the functionalities and requirements of the software products, their size and complexity are also increasing [1]. Usually, the software is an aggregation of a large dataset with thousand of lines of codes. Maintaining the high quality of the software is a critical issue in the practical software development process.

Software defect prediction recommends the potentially defective software entities to software quality assurance teams by mining the historical software development data. It helps the software developers and testers maximize the utilization of the limited resources by allocating them to those risky entities which have the priority to be inspected [2]–[5]. A software entity refers to a method, a file, a package or a change towards the code base [6], [7].

Traditional defect prediction methods first construct classification models based on a sufficient amount of historical labeled software entities from a project, and then use the models to predict the defect labels of new entities within the same project. This setting is referred as Within-Project Defect Prediction (WPDP) [8]–[10]. However, it is not always realistic to obtain sufficient labeled data from a software project, especially for immature or new projects [11], [12]. Meanwhile, manually labeling the unlabeled entities is time-consuming

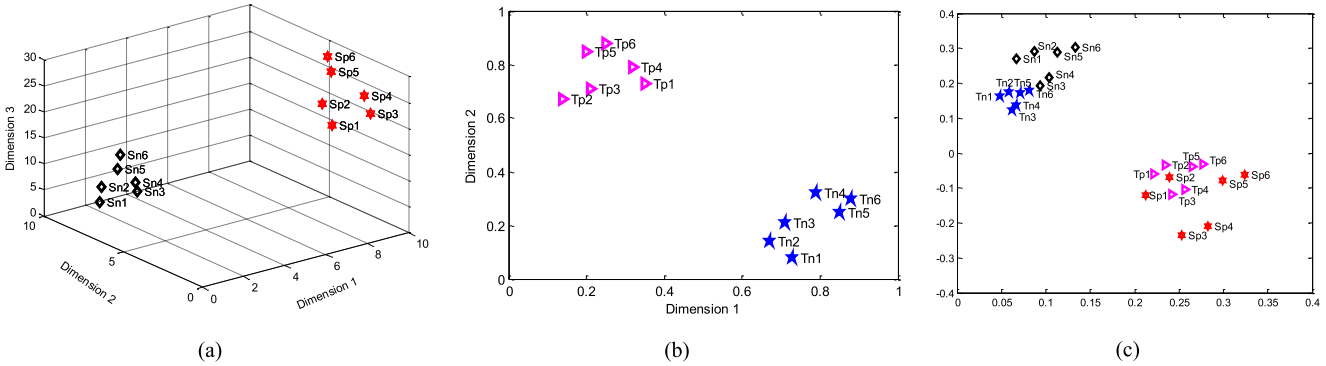


FIGURE 1. An example of domain adaptation. (a) Source domain. (b) Target domain. (c) Embedded space.

and error-prone [11]. In this case, WPDP methods do not work due to the lack of enough labeled data for training effective models.

Fortunately, there is an open source software repository with a large codebase and labeled defect data of other projects. Cross-Project Defect Prediction (CPDP), which constructs classification models based on the labeled defect data from other projects (aka. source projects) to predict the defect labels of the entities from a specific project (aka. target project), emerges to alleviate the issue of defect prediction on scanty labeled data [11]–[14].

However, most existing studies for CPDP assume that the target and source project data have identical features or share some common features. Then, they applied techniques, such as near neighbor filter method [14], weighting-based method [11], and transfer learning method [12], [15], to associate the common features across the two project data, and constructed classification models to perform CPDP. Unfortunately, it is not always feasible that the defect data come from different projects with common features, because different projects may be developed with distinct programming languages and the features may be collected at different levels of granularity using various tools. In this case, traditional CPDP methods fail to perform defect prediction on two project data with completely different feature sets. This setting is referred as Heterogeneous CPDP (HCPDP).

It is non-trivial to accomplish HCPDP because there is no corresponding relationship between the two heterogeneous feature sets which have no common elements. Jing *et al.* [16] and Nam *et al.* [17] were first to raise this issue and proposed methods to address it. However, their methods could not be applied to all heterogeneous cross-project pairs because of some limitations. For example, the method in [17] may not pick out any feature pair across the two project data in the feature matching process if the matching scores of all feature pairs are lower than a given threshold. Therefore, it is important to design a more general method that could work on any cross-project pair with better HCPDP performance.

The difficulty of HCPDP lies in how to bridge the relationship between two different feature spaces to narrow their difference. In this work, we address the challenge from a new perspective of domain adaptation. For the cross-project data, we treat them as being from two domains with different dimensions and heterogeneous feature sets. For example, the first two sub-figures of Fig. 1 depict a 3-dimensional source domain and 2-dimensional target domain, respectively. Sn and Sp denote the negative instances (i.e., non-defective entities) and positive instances (i.e., defective entities) in source domain, respectively. Similarly, Tn and Tp denote the negative and positive instances in the target domain, individually. The two domains share no common features. We aim to seek a consistent representation for the data from the two domains. To this end, we propose a novel Heterogeneous Domain Adaptation (HDA) method based on dictionary learning to embed the data from the two domains into a common feature space with lower dimension and minimize the difference between the two mapped data based on the dictionaries learned from them. As shown in the last sub-figure of Fig. 1, the mapped positive and negative instances of the two domains are connected and comparable within the embedded space.

We conduct substantial experiments to evaluate HDA on 94 heterogeneous cross-project combinations of 12 projects from three open source datasets (i.e., NetGene, NASA, AEEEM datasets) that have heterogeneous feature sets. The data are collected for the real large-scale software systems. Since two previous studies evaluated the effectiveness of the HCPDP methods with F-measure and AUC respectively [16], [17], in this work, we use both indicators and an additional indicator, Balance [10], to comprehensively measure the performance of HDA. We also compare HDA with two HCPDP methods (i.e., CCA+ [16] and HDP [17]), WPDP scenario and a novel unsupervised learning method based on Spectral Clustering (SC). The experimental results indicate that HDA can be applied to any cross-project pair and outperforms the two baseline HCPDP methods in most cases in terms of the three indicators. In addition, compared with WPDP scenario and SC, HDA achieves better or comparable performance in most cross-project pairs.

Our main contributions are highlighted as follows:

1) We propose HDA method for defect prediction on cross-project data with heterogeneous feature sets. HDA employs domain adaptation method to associate the two project data and uses dictionary learning technique to measure the difference between the two mapped project data. HDA can be feasible to any heterogeneous cross-project pair.
2) We empirically perform a comprehensive evaluation for HDA by using three widely-used performance indicators.
3) We conduct extensive experiments on 94 cross-project pairs. The results indicate the superiority of HDA compared with two HCPDP methods and the competitiveness of HDA compared with WPDP scenario and a state-of-the-art unsupervised learning method.

The rest of the paper is organized as follows. Section II introduces the related work. After presenting our method HDA in Section III, we detail the experimental setup and the results in Section IV and Section V, respectively. Section VI discusses the parameter sensitivity to HDA, followed by threats to validity in Section VII. Finally, Section VIII concludes our work.

II. RELATED WORK

A. CROSS-PROJECT DEFECT PREDICTION

For projects without enough labeled data for defect prediction, it is useful to perform CPDP by utilizing the labeled defect data from other projects. This topic has recently attracted many studies.

To our best knowledge, Briand *et al.* [13] were the first to explore whether the defect prediction model built on one system for another system was worth investigating. However, the experimental results on two java systems implied that such model achieved poor performance.

Turhan *et al.* [14] proposed the NN-filter method for CPDP by selecting the source project entities similar to the target project entities. They used the common features of 12 NASA projects to build a cross-project model, and found that the model improved the probability of detecting defects but also dramatically increased the false positive rate.

Zimmermann *et al.* [18] conducted experiments on 622 cross-project pairs with logistic regression model and found that only 3.4% pairs achieve satisfactory performances. Introducing the transfer learning technique to CPDP, Ma *et al.* [11] proposed the TNB method to assign different weights to the source project entities based on the similarity to the target project entities. The experiments showed that TNB achieved better performances than the NN-filter method in [14].

Nam *et al.* [12] proposed the TCA+ method to learn some transfer components for cross-project data in a kernel Hilbert space where their data distributions are similar. The experiments on AEEEM and ReLink datasets showed that TCA+ achieved competitive performances compared with WPDP scenario.

Chen *et al.* [19] proposed a transfer learning method DTB. It first re-weighted target project entities based on data gravitation method and then applied a transfer boosting method to decrease the weights of negative entities in target project data. The experiments showed that DTB with limited labeled target project data outperformed four baseline CPDP methods and achieved better performances than three WPDP methods.

However, all these studies assume that the target and source project data share some common features. Therefore, if the cross-project data have heterogeneous feature sets (i.e., HCPDP scenario), these methods are failure.

B. HETEROGENEOUS CROSS-PROJECT DEFECT PREDICTION

Jing *et al.* [16] proposed the CCA+ method for HCPDP. CCA+ first used Unified Metric Representation (UMR) technique to unify the two heterogeneous feature sets and then applied Canonical Correlation Analysis (CCA) technique to maximize the similarity of the two unified feature sets. On one hand, CCA+ needs to make the dimensions of the two feature sets the same by complementing extra features with zero values. However, if adding too many such features, it will make the covariance matrices singularity during the process of solving CCA. Consequently, we may obtain transformed target project data whose features are all zero values for some cross-project pairs. CCA+ fails to perform HCPDP in such cases. On the other hand, CCA+ needs to calculate the cross-project covariance matrix $C_{ST} = \frac{1}{n\times m}\sum_{i=1}^{n}\sum_{j=1}^{m}(\overline{x}_S^i - m_S)(\overline{x}_T^i - m_T)^{\mathrm{T}}$, where n and m denote the number of source and target project entities, respectively. $\overline{x}_S^i$ and $\overline{x}_T^i$ represent the ith and jth entity vector of the unified source data matrices $\overline{X}_S$ and $\overline{X}_T$, respectively. m_S and m_T indicate the mean values of the feature vectors of $\overline{X}_S$ and $\overline{X}_T$, individually. From the definition, we find that each element of C_{ST} will be extremely small due to being divided by $n \times m$, especially when n and m are large. Besides, calculating C_{ST} will be very slow due to the double sum for each feature of $\overline{X}_S$ and $\overline{X}_T$, especially when the feature dimension is large. These limitations will make CCA+ inapplicable to some cross-project pairs and time-consuming, especially for defect data with many features and entities.

Nam *et al.* [17] proposed the HDP method for HCPDP. This method first used feature selection techniques to eliminate the useless features in the source project data and then used the similarity calculating methods to match the selected features in the source project data with the features in the target project data and remove the matched features whose matching scores are lower than a given threshold. However, when all the scores are lower than the threshold, HDP fails to perform HCPDP since we may not obtain any matched features. Hence, HDP is also not appropriate for some cross-project pairs. Our new HDA method has no constraints on the feature dimensions for the cross-project data, thus we do not need to add useless features with all zero values. In addition, HDA utilizes all features of the two

project data without eliminating any feature. Therefore, HDA is more general than the two above methods for handling heterogeneous cross-project pairs.

Recently, Kong and Wang [20] proposed a novel method to transfer the unlabeled data from the source domain to the heterogeneous target domain and used a dictionary-based instance selection method to transfer the most informative instances in the source domain. In this work, we adopt the first part of the method in [20] to establish the association of the target and source project data for HCPDP task. Different from the work in [20] aiming at unsupervised clustering, we use the method for heterogeneous defect classification. We refer the method as Heterogeneous Domain Adaptation (HDA).

C. HETEROGENEOUS DOMAIN ADAPTATION

Domain adaptation aims to transfer knowledge from a source domain to a different but related target domain. The key is to learn a good feature representation for the two domains. Domain adaptation methods have been successfully applied to different research fields, such as remote sensing classification [21], [22] and computer vision [23], [24]. However, traditional domain adaptation methods assume that the data from the two domains are represented by identical features with the same dimension [25]. Therefore, they cannot be directly applied to the scenario where the two data domains have different dimensions. This scenario is referred as heterogeneous domain adaptation [26]. The basic idea of addressing this problem is to use two mapped matrices to transform the data from the two domains into a lower-dimensional feature space where the similarity of the two mapped data can be measured. Shi *et al.* [27] proposed the HeMap method to embed the data from two domains into a consistent space where the structures of original data were preserved and the similarity of the two mapped data was maximized. The limitation of HeMap is that it needs to make the instance number of two domains the same with a random sampling technique, which may alter the data distributions of original data. Harel and Mannor [28] learned optimal affine mapping matrices to match the empirical distributions of two heterogeneous domains. Kong and Wang [20] transferred the unlabeled data from the source domain to the heterogeneous target domain and used a dictionary-based instance selection method to transfer the most informative instances in the source domain. Here, we adopt the first part of the method in [20] to establish the association of the target and source project data for HCPDP task. Different from the work in [20] aims at unsupervised clustering, we use the method for heterogeneous defect classification. We refer the method as Heterogeneous Domain Adaptation (HDA).

III. METHOD

A. PROBLEM DEFINITION

Let the target project data matrix be $\boldsymbol{T} = [t_1, t_2, ..., t_m]$, where $t_i = [t_{i1}, t_{i2}, ..., t_{ip}]^{\mathrm{T}} \in \mathbb{R}^p$. m and p denote the number of target project entities and features, respectively. T_{ij} indicates the jth feature of the ith target project entity. Similarly, let the source project data matrix be $\boldsymbol{S} = [s_1, s_2, ..., s_n]$, where $s_i = [s_{i1}, s_{i2}, ..., s_{iq}]^{\mathrm{T}} \in \mathbb{R}^q$. n and q denote the number of source project entities and features, individually. S_{ij} indicates the jth feature of the ith source project entity. Due to the large differences of value ranges between distinct features, we use the z-score technique [29] to normalize the two project data.

We treat the target and source project data as being from two heterogeneous domains with different feature spaces, i.e., $\mathbb{R}_p \neq \mathbb{R}_q$. To transfer the knowledge from the source project domain to adapt the heterogeneous target project domain, we seek for two mapped matrices $\boldsymbol{W_T} \in \mathbb{R}^{p\times k}$ and $\boldsymbol{W_S} \in \mathbb{R}^{q\times k}$(where k is the finally selected feature number of the mapped data) for the target and source project data, and map them into a lower-dimensional feature space (i.e., $k \leq \min(p, q)$) where the intrinsic similarity among the initial source and target project data are preserved and the two mapped data are comparable. Since the number of software entities in the two project data is different, to measure the difference between the two mapped data, we learn two dictionaries $\boldsymbol{D_T} = [d_{T1}, d_{T2}, ..., d_{Td}] \in \mathbb{R}^{k\times d}$, $\boldsymbol{D_S} = [d_{S1}, d_{S2}, ..., d_{Sd}] \in \mathbb{R}^{k\times d}$ for the mapped target and source project data, respectively. d_{Ti} (or d_{Si}) denotes the ith dictionary atom of $\boldsymbol{D_T}$ (or $\boldsymbol{D_S}$), and d indicates the desired size of the dictionary (i.e., the number of the atom). Then, we use $\|\boldsymbol{D_T} - \boldsymbol{D_S}\|_2$ to measure the difference between the two dictionaries, where $\|\cdot\|_2$ denotes l_2-norm. $\|\boldsymbol{D_T} - \boldsymbol{D_S}\|_2$ can also be used to measure the difference between the two mapped data because the two dictionaries are individually learned from them. To reduce the difference between the mapped data and its sparse representation by dictionary learning and the difference between the two learned dictionaries, we minimize the following objective function $G(\boldsymbol{W_T}, \boldsymbol{W_S}, \boldsymbol{D_T}, \boldsymbol{D_S}, \boldsymbol{A_T}, \boldsymbol{A_S})$ that consists of three difference functions(i.e., those in parentheses):

$$\begin{aligned} &G(\boldsymbol{W_T}, \boldsymbol{W_S}, \boldsymbol{D_T}, \boldsymbol{D_S}, \boldsymbol{A_T}, \boldsymbol{A_S}) \\ &= (\|\boldsymbol{W_T^{\mathrm{T}}T} - \boldsymbol{D_T A_T}\|_2 \\ &\quad + \gamma\|\boldsymbol{A_T}\|_1 + \eta\|\boldsymbol{W_T}\|_2) + (\|\boldsymbol{W_S^{\mathrm{T}}S} - \boldsymbol{D_S A_S}\|_2 + \gamma\|\boldsymbol{A_S}\|_1 \\ &\quad + \eta\|\boldsymbol{W_S}\|_2) + (\lambda\|\boldsymbol{D_T} - \boldsymbol{D_S}\|_2) \end{aligned} \tag{1}$$

where $\|\cdot\|_1$ demnotes l_1-norm, $\boldsymbol{A_T} \in \mathbb{R}^{d\times m}$ and $\boldsymbol{A_S} \in \mathbb{R}^{d\times n}$ represent the sparse coefficient matrices. $\|\boldsymbol{W_T^{\mathrm{T}}T} - \boldsymbol{D_T A_T}\|_2$ measures the difference between the mapped target project data and its sparse representation. Similarly, $\|\boldsymbol{W_S^{\mathrm{T}}S} - \boldsymbol{D_S A_S}\|_2$ measures the difference between the mapped source project data and its sparse representation. γ controls the sparsity of the coefficient matrices $\boldsymbol{A_T}$ and $\boldsymbol{A_S}$. η determines the stability of the solution of $\boldsymbol{W_T}$ and $\boldsymbol{W_T}$ by avoiding the two mapped matrices too large. λ controls the difference between the two learned dictionaries $\boldsymbol{D_T}$ and $\boldsymbol{D_S}$. This objective function bridges the relationship between the two heterogeneous data domains.

B. SOLUTION STEPS

Equation (1) is a multi-variable optimization problem, we solve it following the five steps below.

Step 1 (Initialize the Mapped Matrices and the Dictionaries): We initialize the mapped matrices $\boldsymbol{W_T}$ and $\boldsymbol{W_S}$ as the first k left vectors of target project data matrix $\boldsymbol{T}$ and source project data matrix $\boldsymbol{S}$, respectively. Then, we employ a dictionary learning algorithm k-SVD [30] on the mapped target project data $\boldsymbol{W_T}^{\mathrm{T}}\boldsymbol{T}$ and and the mapped source project data $\boldsymbol{W_S}^{\mathrm{T}}\boldsymbol{S}$ to initialize the dictionaries $\boldsymbol{D_T}$ and $\boldsymbol{D_S}$.

Step 2 (Update the Sparse Coefficient Matrices): We update the two sparse coefficient matrices $\boldsymbol{A_T}$ and $\boldsymbol{A_S}$ one by one by fixing other four parameters (i.e., $\boldsymbol{W_T}$, $\boldsymbol{W_S}$, $\boldsymbol{D_T}$ and $\boldsymbol{D_S}$) and ignoring the unrelated terms towards the coefficient matrices. For example, to update $\boldsymbol{A_T}$, we solve the following objective:

$$\boldsymbol{A_T} = \underset{\boldsymbol{A_T}}{arg\,min}\,(\|\boldsymbol{W_T}^{\mathrm{T}}\boldsymbol{T} - \boldsymbol{D_T}\boldsymbol{A_T}\|_2 + \gamma\|\boldsymbol{A_T}\|_1) \tag{2}$$

This is a LASSO problem [31] that can be solved with feature-sign search algorithm [32]. We update $\boldsymbol{A_S}$ in the same way.

Step 3 (Update the Dictionaries): We update the dictionary by updating its atoms one by one. Since the two dictionaries, $\boldsymbol{D_T}$ and $\boldsymbol{D_S}$, are associated with the term $\|\boldsymbol{D_T} - \boldsymbol{D_S}\|_2$, we update one by fixing the other and ignoring its unrelated terms. For example, given $\boldsymbol{A_T} = [a_{T1}, a_{T2}, ..., a_{Td}]$, to update the jth atom d_{Tj} of $\boldsymbol{D_T}$, we minimize the following formula:

$$g = \|\boldsymbol{W_T}^{\mathrm{T}}\boldsymbol{T} - \sum_{i\neq j} d_{Ti}a_{Ti} - d_{Tj}a_{Tj}\|_2 + \lambda\|d_{Tj} - d_{Sj}\|_2 \tag{3}$$

The updated atom d_{Tj} is obtained by setting the first-order derivative in terms of d_{Tj} as 0, i.e., $\partial g/\partial d_{Tj} = 0$. We update $\boldsymbol{D_S}$ in the same way.

Step 4 (Update the Mapped Matrices): We update one mapped matrix by ignoring its unrelated terms. For example, to update $\boldsymbol{W_T}$, we solve the following objective:

$$\boldsymbol{W_T} = \underset{\boldsymbol{W_T}}{arg\,min}\,(\|\boldsymbol{W_T}^{\mathrm{T}}\boldsymbol{T} - \boldsymbol{D_T}\boldsymbol{A_T}\|_2 + \eta\|\boldsymbol{W_T}\|_2) \tag{4}$$

According to the regularized least squares algorithm [33], (4) can be rewritten as:

$$\boldsymbol{W_T} = (\boldsymbol{T}\boldsymbol{T}^{\mathrm{T}} + \eta\boldsymbol{I})^{-1}\boldsymbol{T}\boldsymbol{A_T}^{\mathrm{T}}\boldsymbol{D_T}^{\mathrm{T}} \tag{5}$$

We update $\boldsymbol{W_S}$ in the same way.

Step 5 (Iteratively Solving): Go back to step 2 to iteratively update $\boldsymbol{A_T}$, $\boldsymbol{A_S}$, $\boldsymbol{D_T}$, $\boldsymbol{D_S}$, $\boldsymbol{W_T}$, $\boldsymbol{W_S}$ until the difference of function $G(\boldsymbol{W_T}, \boldsymbol{W_S}, \boldsymbol{D_T}, \boldsymbol{D_S}, \boldsymbol{A_T}, \boldsymbol{A_S})$ between two consecutive iterations is small enough or the maximum number of iteration reaches.

Finally, we obtain the optimal parameters $\boldsymbol{A_T}, \boldsymbol{A_S}, \boldsymbol{D_T}, \boldsymbol{D_S}$, $\boldsymbol{W_T}, \boldsymbol{W_S}$, the mapped target project data $\boldsymbol{B_T} = \boldsymbol{W_T}^{\mathrm{T}}\boldsymbol{T}$ and the mapped source project data $\boldsymbol{B_S} = \boldsymbol{W_S}^{\mathrm{T}}\boldsymbol{S}$.

IV. EXPERIMENTAL SETUP

A. BENCHMARK DATASETS

We use three public open-source software defect datasets as our benchmarks, including NetGene, NASA, AEEEM.

NetGene dataset, collected by Herzig *et al.* [34], was derived from more than 7400 issue reports by manual inspection and the quality is manually verified. NASA dataset, the most popular benchmark dataset for defect prediction, was collected from the systems of NASA contractors [10]. We select four projects that share the same features from this dataset. AEEEM dataset was collected by D'Ambros *et al.* [35]. Features in this dataset include the change metrics, source code metrics, entropy of source code metrics, and churn of source code metrics.

TABLE 1. Statistic of benchmark datasets.

Datasets	Projects	Size	# Features	# Entities	% Defective	Granularity
NetGene	Httpclient (H)	140678	466	361	56.79%	Change
	Lucene (L)	3260957	466	1671	20.71%	
	Rhino (R)	88971	466	253	43.08%	
NASA	cm1	15318	37	327	12.84%	Function
	mw1	6793	37	253	10.67%	
	pc3	32226	37	1077	12.44%	
	pc4	30055	37	1458	12.21%	
AEEEM	EQ	39534	61	324	39.81%	Class
	JDT	224055	61	997	20.66%	
	LC	73184	61	691	9.26%	
	ML	156102	61	1862	13.16%	
	PDE	146952	61	1497	13.96%	

Table 1 reports the details of the projects in the three datasets, including the size of the project (Size), number of features (# Features), the number of entities (# Entities), the percentage of defective entities (% Defective) and the granularity of the features. For NetGene dataset, the size of the project is defined as the number of the changed files, whereas for other two datasets, the size of the project is defined as the number of the lines of code. The projects in the three datasets are relatively large softwares since the units of these projects vary from 6793 to 3260957. All these three datasets are widely used for defect prediction [12], [16], [17], [34]–[40].

B. PERFORMANCE INDICATORS

To evaluate the performance of HDA for HCPDP, we employ the following three indicators, namely F-measure, AUC and Balance, which are widely used in defect prediction.

1) F-measure. For a binary classification model, we can obtain four outcomes on test entities: the number of defective entities that classified correctly ($n_{d\to d}$) or incorrectly ($n_{d\to f}$), the number of defect-free entities that classified correctly ($n_{f\to f}$) or incorrectly ($n_{f\to d}$). The defective precision $Pre = n_{d\to d}/(n_{d\to d} + n_{f\to d})$, defective recall $Rec = n_{d\to d}/(n_{d\to d} + n_{d\to f})$. The defective F-measure is a harmonic mean of Pre and Rec, which is defined as:

$$F\text{-}measure = \frac{(1+\theta^2)\times Pre \times Rec}{\theta^2 \times Pre + Rec} \tag{6}$$

where θ is a bias parameter towards precision and recall. There are three extensively-used F-measure, including F_1 ($\theta = 1$) which treats precision and recall equally,

$F_{0.5}$ ($\theta = 0.5$) which prefers precision, and F_2 ($\theta = 2$) which prefers recall. Since recall is more important from the defect prediction point of view [41], we choose F_2 as our performance indicator F-measure following the previous work [42].

2) Balance. The point (Pf, Pd)=(0,1) is the perfect position on the ROC curve, which indicates that all the defective entities are identified correctly. Menzies *et al.* [10] proposed a new indicator, called Balance, to evaluate the performance of defect prediction. Balance measures the normalized Euclidean distance from the current point (Pf, Pd) to the perfect point (0,1), which is defined as:

$$Balance = 1 - \sqrt{\frac{(0 - pf)^2 + (1 - pd)^2}{2}} \tag{7}$$

This indicator is widely used in defect prediction [43]–[45].

3) AUC. ROC is a curve plotted on a two-dimensional plane with the true positive rate ($Pd = Rec$) as the y-axis and the false positive rate ($Pf = n_{f\to d}/(n_{f\to d} + n_{f\to f})$) as the x-axis. This curve is used to visualize the performance of binary classifiers. To quantitatively evaluate the performance through the curve, it is common to calculate the relative Area Under the Curve (so-called AUC) as a single scalar value [46].

Higher F-measure, Balance and AUC values indicate better defect prediction performance.

C. PREDICTION MODEL

In this work, we train a logistic regression model [47] with the mapped source project data $\boldsymbol{B_S}$ and then apply it to the mapped target project data $\boldsymbol{B_T}$. The logistic regression model is widely used for defect prediction [48]–[56].

D. EXPERIMENTAL DESIGN

In step 1 of HDA, the k-SVD algorithm is used to learn the dictionaries. k-SVD initializes the atoms of $\boldsymbol{D_T}$ and $\boldsymbol{D_S}$ by randomly selecting the entities, and the randomness may impact the experimental results. To alleviate the bias, we run HDA 30 times and report the average indicator values. For the parameter values, we empirically set $\gamma = 0.5$, $\lambda = 5$, $\eta = 0.001$, $k = 5$ as the basic setting through extensive experiments. We will discuss the impact of different parameter settings on the experimental results in Section VI.

V. EXPERIMENTAL RESULTS

To evaluate the effectiveness of HDA, we investigate the following two research questions (in Section V-A and V-B).

A. RQ1: HOW DOES HDA PERFORM FOR HCPDP?

1) MOTIVATION

It is challenging to establish the relationship between two completely different feature sets for HCPDP. We compare our method HDA with two state-of-the-art methods (i.e., CCA+ [16] and HDP [17]) in terms of their performances in solving HCPDP.

2) APPROACH

Since two recent studies [16], [17] used F-measure and AUC to evaluate the effectiveness of their HCPDP methods respectively, we use both of them and include a new indicator (i.e., Balance). Since the projects from one dataset have identical feature set, to simulate HCPDP scenario, we select the source project and the target project from different datasets. Thus, we obtain total 94 (i.e., $3 \times (4 + 5) + 4 \times (3 + 5) + 5 \times (3 + 4) = 94$) heterogeneous cross-project combinations. Since both CCA+ and HDP do not involve any randomness, we run them only once and report the results. For CCA+, as the authors did not state the number of features selected for the mapped target and source project data, to make a fair comparison, we choose the features as the eigenvectors corresponding to the top five largest eigenvalues. For HDP, it uses four feature selection methods in feature selection phase, three similarity calculating methods and ten thresholds in feature matching phase. We choose the combination as 'significance attribute selection' method for feature selection and 'Kolmogorov-Smirnov Test' with the threshold of 0.05 for feature matching which achieved the best performance as shown in [17].

3) RESULTS

Fig. 2, 3, and 4 show the F-measure, Balance, AUC for the three methods on each heterogeneous cross-project pair, respectively. Different colors represent distinct indicator values and the darker color indicates better performance. The project in the vertical axis represents the target project and the project in the horizontal axis represents the source project. For the projects in the same dataset, we do not conduct cross-project prediction among them and assign the corresponding indicator value with -1 which is represented by the dark blue color (i.e., the grids near the diagonal in the figure). Note that all the ranges of the three indicators used in this work are between 0 and 1. For the CCA+ method, when it is applied to some cross-project pairs, the features in the mapped target project data are all 0 values. In such case, we treat the CPDP process as a failure, and assign the corresponding indicator values with 0. For HDP method, when the matching scores of all feature pairs are lower than the threshold, we cannot select any features because all feature pairs are eliminated. In such case, we also treat the CPDP process as a failure and set the corresponding indicator values as 0. From the experimental results and the three figures, we observe that:

First, according to the detailed results, when the projects in NetGene dataset act as the target project, HDA and CCA+ can handle all cross-project pairs, whereas HDP fails on 12 pairs (showed as the white cells). When the projects in NASA dataset serve as the target projects, our method HDA can be applied to all cross-project pairs, whereas CCA+ and HDP fail on 17 and 25 cross-project pairs, respectively. When the projects in AEEEM dataset are the target projects, HDA can handle all cross-project pairs, whereas HDP fails on 23 cross-project pairs and CCA+ fails on all pairs since

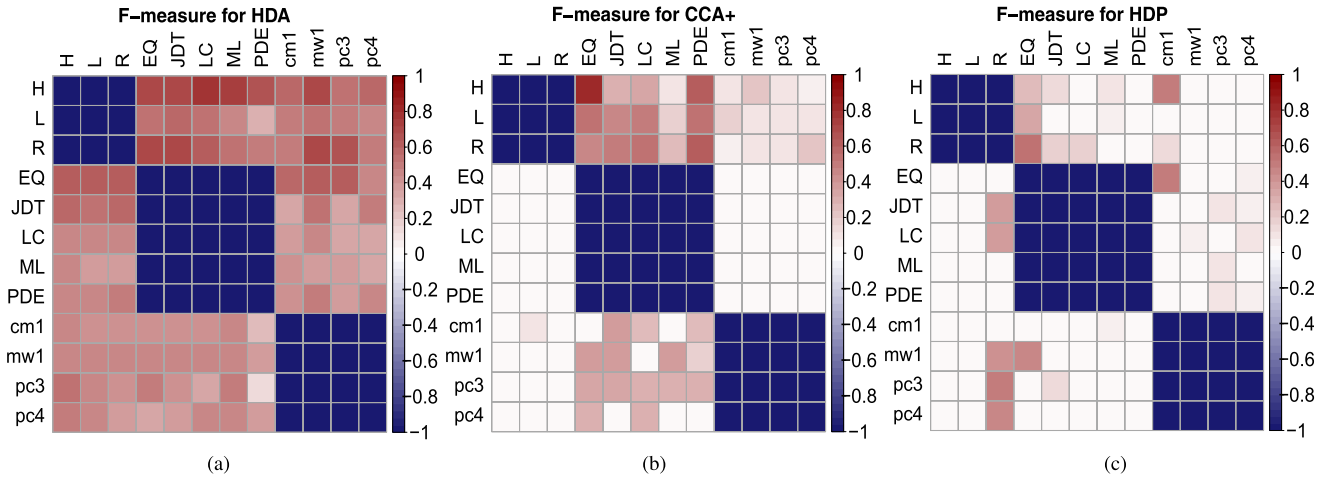


FIGURE 2. F-measure of our HDA method and the two baseline methods on each heterogeneous cross-project pair. (a) F-measure for HDA. (b) F-measure for CCA+. (c) F-measure for HDP.

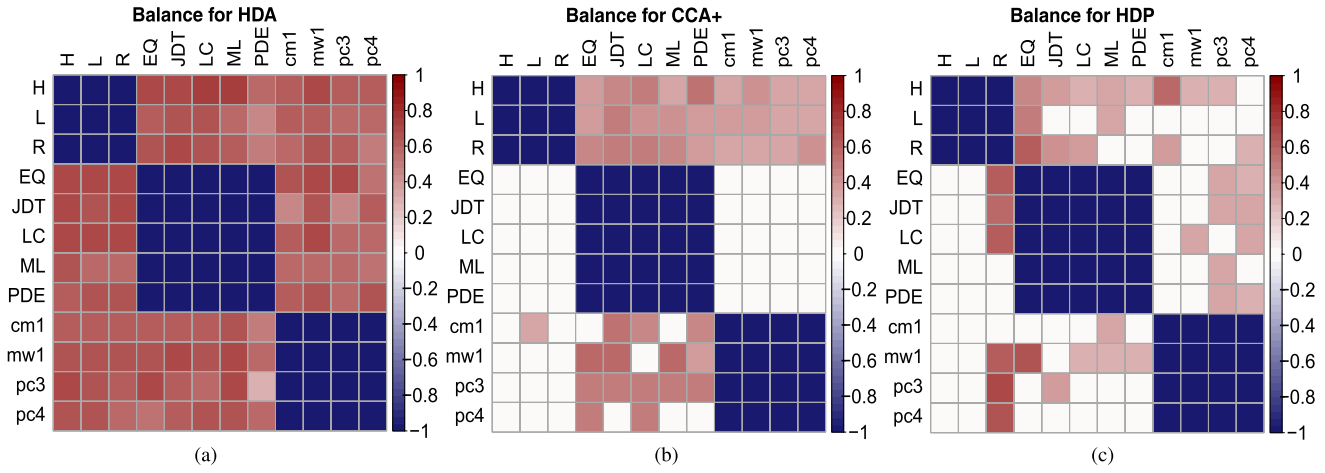


FIGURE 3. Balance of our HDA method and the two baseline methods on each heterogeneous cross-project pair. (a) Balance for HDA. (b) Balance for CCA+. (c) Balance for HDP.

the mapped target project features with all 0 values in all cases.

Second, from the Fig. 2, in terms of F-measure, when the projects in NetGene dataset act as the target project, HDA outperforms CCA+ and HDP on 23 and 27 cross-project pairs among total 27 pairs, respectively. When the projects in NASA dataset serve as the target project, HDA outperforms CCA+ and HDP on 31 and 30 cross-project pairs among total 32 pairs, respectively. When the projects in AEEEM dataset are the target project, HDA outperforms HDP on all 35 cross-project pairs.

Third, from the Fig. 3, in terms of Balance, when the projects in NetGene dataset act as the target project, HDA outperforms CCA+ and HDP on all 27 cross-project pairs. When the projects in NASA dataset serve as the target project, HDA outperforms CCA+ and HDP on 31 and 30 cross-project pairs among total 32 pairs, respectively. When the projects in AEEEM dataset are the target project, HDA outperforms HDP on all 35 cross-project pairs.

Fourth, from the Fig. 4, in terms of AUC, when the projects in NetGene dataset act as the target project, HDA outperforms CCA+ and HDP on 26 and 23 cross-project pairs among total 27 pairs, respectively. When the projects in NASA dataset serve as the target project, HDA outperforms CCA+ and HDP on 31 and 29 cross-project pairs among total 32 pairs, respectively. When the projects in AEEEM dataset are the target project, HDA outperforms HDP on 30 cross-project pairs among total 35 pairs.

Fifth, for our HDA method, we observe that the results of a specific target project with different source projects have no big difference in most cases since the cell color in each row are very similar. For example, when project mw1 is the target project, the F-measure, Balance, and AUC values are around 0.460, 0.670, and 0.730 respectively under different heterogeneous source projects, with the exception when PDE act as the source project (i.e., the eighth column). It implies that HDA can achieve stable performance towards a target project with different heterogeneous source projects.

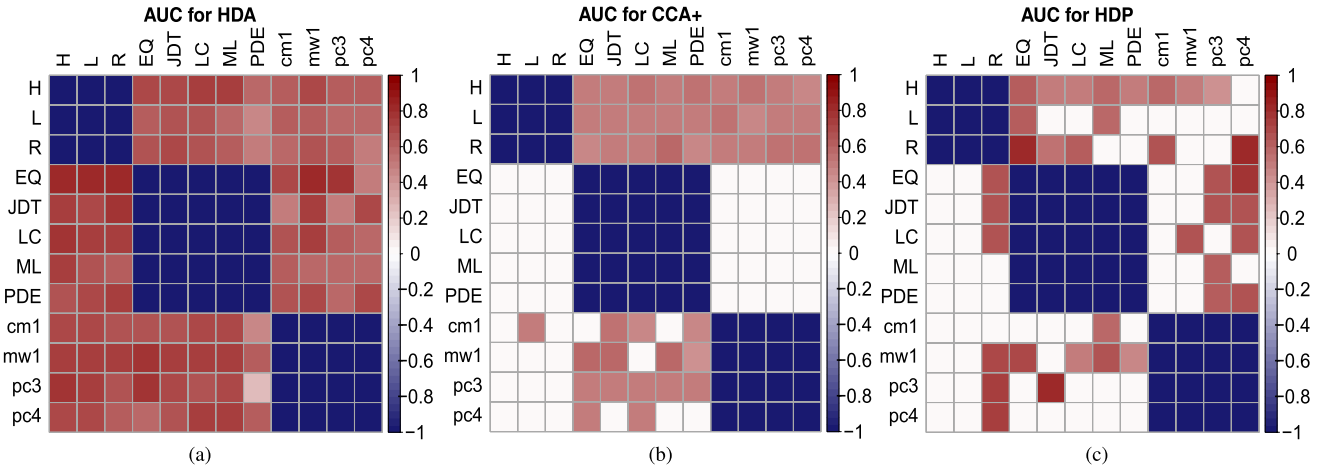


FIGURE 4. AUC of our HDA method and the two baseline methods on each heterogeneous cross-project pair. (a) AUC for HDA. (b) AUC for CCA+. (c) AUC for HDP.

In summary, HDA can be applicable to any cross-project pairs and has the potential to achieve encouraging results compared with two HCPDP methods.

B. RQ2: CAN HDA ACHIEVE COMPARABLE PERFORMANCE COMPARED WITH WPDP AND THE UNSUPERVISED LEARNING METHOD?

1) MOTIVATION

In WPDP scenario, some labeled entities from the target project are used to train a prediction model and the resultant model is used to predict the class labels of the other entities. Previous studies have showed that the performance of WPDP will be improved if there are sufficient labeled training data available [18]. When a project has scanty training data, CPDP is an alternative way for defect prediction. However, existing studies have pointed out that CPDP methods usually do not perform well compared with WPDP in most cases [18], [57]. For HCPDP, the experiments in [16] showed that CCA+ achieved better F-measure than WPDP on 12 cross-project pairs among total 90 cross-project pairs. Thus, we are interested in whether our method HDA can achieve better or comparable results than WPDP in terms of the three performance indicators. In addition, another way to perform defect prediction on project with scanty labeled data is to apply unsupervised learning methods, such as the clustering-based methods, to partition the software entities into different clusters and label each cluster.

However, previous studies have shown that distance based unsupervised learning method achieved disappointing results [58]. Recently, Zhang *et al.* [59] proposed a novel Spectral Clustering (SC) based unsupervised method for defect prediction without the need of source projects. Different from distance-based unsupervised learning method, SC performs on a graph with nodes and edges to divide the data points into different clusters. In defect prediction, the node represents the software entity and the edge presents the similarity of the feature values between the two nodes of the edge. The experimental results have shown that SC method achieved impressive AUC values compared with five supervised learning methods, four distance-based unsupervised learning methods and WPDP. However, the main limitation of SC is that it needs a heuristic to determine the labels of the software entities in different clusters. Once the heuristic is wrong, all the entities will be misclassified. Here, we are also interested in investigating whether our HDA method can achieve better or comparable performance with the aid of heterogeneous data compared with SC.

2) APPROACH

For WPDP, we randomly select 50% entities in the target project as the training set and the remaining entities as the test set. This setting follows the previous studies [16], [60], [61]. With respect to the classification model used for WPDP, we investigate the performance of five widely-used classifiers, including Logistic Regression (LR), Naive Bayes (NB), k-Nearest Neighbor (kNN), Random Forest (RF), Classification And Regression Tree (CART). We employ two types of NB classifier with different distribution settings: Normal distribution (or Gaussian distribution) and Kernel distribution. We abbreviate the two classifiers as NB_N and NB_K, respectively. For the parameter k in kNN classifier, prior studies [16] and [62] individually suggested it as 1 and 8. Hence, we use both parameters for kNN classifier, short for NN_1 and NN_8, respectively. For other three classifiers, we use the default parameter values. Since the prediction performance may be affected by the random selection process for the training set and test set, we repeat the process 30 times to obtain the average indicator values. For SC, we use the code provided by the authors to reproduce this method.

3) RESULTS

Through the experiments, we observe that, on the 12 projects, WPDP with NB_K classifier can achieve the best results for

TABLE 2. Three indicator values of HDA, WPDP and SC in some specific heterogeneous cross-project pairs.

Cross-Pairs		F-measure			Balance			AUC		
Source	Target	HDA	WPDP	SC	HDA	WPDP	SC	HDA	WPDP	SC
EQ	H	0.716	0.907	0.800	0.714*	0.436	0.750	**0.755***	0.554	0.743
JDT		0.712			0.706*			**0.734***		
LC		**0.791**			**0.745***			0.717*		
ML		0.721			**0.737***			**0.814***		
mw1		0.709			0.684*			0.689*		
EQ	L	0.523	0.892	0.592	**0.636***	0.463	0.649	**0.672***	0.519	0.681
JDT		**0.581**			**0.652***			**0.697***		
LC		0.545			**0.642***			**0.668***		
mw1		0.529			**0.638***			0.652*		
EQ	R	**0.712**	0.893	0.676	0.666*	0.294	0.738	0.752*	0.484	0.826
JDT		**0.683**			0.703*			0.770*		
mw1		**0.705**			0.646*			0.727*		
pc3		**0.664**			0.632*			0.709*		
H	EQ	**0.605**	0.755	0.591	**0.682***	0.443	0.671	**0.828***	0.611	0.800
L		**0.613**			**0.687***			**0.805***		
R		**0.607**			**0.684***			**0.807***		
mw1		**0.638**			**0.701***			**0.809***		
pc3		**0.623**			**0.689***			**0.789***		
H	JDT	0.582	0.610	0.650	0.690*	0.574	0.753	0.738*	0.679	0.830
R		0.595*			0.706*			0.779*		
mw1		0.551			0.674*			0.738*		
H	LC	**0.464***	0.335	0.488	0.702*	0.488	0.725	0.760*	0.599	0.790
L		0.450*			0.687*			0.752*		
R		0.458*			0.694*			0.756*		
mw1		0.445*			0.686*			0.735*		
H	ML	**0.46***	0.375	0.408	**0.646***	0.513	0.606	**0.745***	0.586	0.630
L		0.371*			0.574*			**0.642***		
R		**0.398***			**0.597***			**0.622***		
cm1		**0.407***			**0.598***			**0.613***		

Cross-Pairs		F-measure			Balance			AUC		
Source	Target	HDA	WPDP	SC	HDA	WPDP	SC	HDA	WPDP	SC
pc3	PDE	**0.391***	0.464	0.493	0.570*	0.545	0.669	0.580*	0.594	0.720
H		0.444*			0.629*			0.672*		
L		0.459*			0.642*			**0.700***		
R		**0.488***			**0.665***			**0.720***		
mw1		**0.482***			**0.660***			**0.704***		
pc4		0.459*			0.640*			0.686		
EQ	cm1	**0.415***	0.374	0.433	**0.606***	0.556	0.628	**0.672***	0.633	0.670
JDT		0.406*			**0.607***			**0.674***		
LC		**0.411***			**0.610***			**0.688***		
ML		**0.475***			**0.660***			**0.711***		
H		**0.446***			**0.638***			**0.706***		
L		**0.437***			**0.629***			**0.699***		
R		0.405*			**0.605***			**0.658***		
EQ	mw1	**0.466***	0.432	0.466	**0.674***	0.654	0.685	**0.764***	0.706	0.700
JDT		**0.466***			**0.681***			**0.743***		
LC		**0.454***			**0.673***			**0.738***		
ML		**0.469***			**0.683***			**0.724***		
H		**0.463***			**0.677***			**0.738***		
L		**0.461***			**0.672***			**0.721***		
R		**0.463***			**0.675***			**0.737***		
EQ	pc3	**0.515***	0.496	0.487	**0.695***	0.514	0.680	**0.767***	0.581	0.720
JDT		0.438			0.639*			**0.705***		
ML		**0.512***			**0.689***			**0.718***		
H		**0.545***			**0.718***			**0.767***		
L		**0.477***			**0.668***			**0.729***		
LC	pc4	**0.459**	0.530	0.477	**0.659***	0.607	0.674	**0.720***	0.686	0.732
ML		**0.475**			**0.671***			**0.723***		
H		**0.488**			**0.672***			0.700*		
L		**0.462**			0.648*			0.690*		

the three indicators in most cases. Thus, we only report the indicator values of this classifier. Table 2 reports the three indicator values of HDA, WPDP, and SC in some specific cross-project cases. The indicator values of HDA are in bold font and with an asterisk (*) if they are better or comparable (i.e., the difference less than 0.02) compare with that of SC and WPDP, respectively. From the table, we observe that:

First, in terms of F-measure, WPDP can achieve the best results when the projects in NetGene act as the target projects. Compare with the two baseline methods, our method HDA can obtain better or comparable results when the projects in AEEEM and NASA serve as the target project in most cases.

Second, in terms of Balance and AUC, our method HDA outperform WPDP in all the presented cases (as the values with an asterisk). In addition, HDA achieve better or comparable results than SC in most cases (as the values in bold font).

Third, SC outperform WPDP on 11 out of 12 projects in terms of AUC. This observation is consistent with the conclusion in [59] that SC is competitive compared with some WPDP methods. In addition, SC outperforms WPDP on all projects in terms of Balance, but WPDP achieve the best F-measure than SC on the three projects of NetGene dataset.

Fourth, to statistically analyze the performance values in the table among the three methods, we perform the non-parametric Frideman test with the Nemenyi's post-hoc test [63] at significant level 0.05 over the cross-project pairs. The Friedman test evaluates whether there exist statistically significant differences among the average ranks of different methods. Since Friedman test is based on performance ranks of the methods, rather than actual performance values, therefore it makes no assumptions on the distribution of performance values and is less susceptible to outliers [48], [64]. The test statistic of the Friedman test can be calculated as follows:

$$\tau_{\chi^2} = \frac{12N}{L(L+1)}\left(\sum_{j=1}^{L} AR_j^2 - \frac{L(L+1)^2}{4}\right), \tag{8}$$

where N denotes the total number of the cross-project pairs, L denotes the number of methods needed to be compared, $AR_j = \frac{1}{N}\sum_{i=1}^{N} R_i^j$ denotes the average rank of method j across all cross-project pairs and R_i^j denotes the rank of jth method on the ith cross-project pair. τ_{χ^2} obeys the χ^2 distribution with $L-1$ degree of freedom.

If the null hypothesis is rejected, it means that the performance differences among the methods are nonrandom, then a so-called Nemenyi's post-hoc test is performed to check which specific method differs significantly [65]. For each pair of the methods, this test uses the average rank of each method and checks whether the rank difference exceeds a Critical Difference (CD) which is calculated with the following formula:

$$CD = q_{\alpha,L}\sqrt{\frac{L(L+1)}{6N}}, \tag{9}$$

where $q_{\alpha,L}$ is a critical value that related to the number of methods L and the significance level α. The Frideman test with the Nemenyi's post-hoc test is widely used in previous studies [36], [48], [64]–[68]

Fig. 5 visualizes the results of the Friedman test with Nemenyi's post-hoc test for the three methods in terms of the three indicators. Methods that are significantly different are not connected. The results of the Friedman test show that the p values on all three indicators are all less than 0.05, which means that there exist significant differences among the 3 methods in terms of all indicators. The results of the post-hoc test show that in terms of F-measure, SC performs no significant difference compared with WPDP while WPDP performs no significant difference compared with HDA. In terms of Balance, SC performs significant difference than other two methods while our method HDA performs significant difference than WPDP. In terms of AUC, SC and HDA perform no significant difference with each other but both perform significant difference compared with WPDP. Since AUC is independent of the classifier threshold, it is

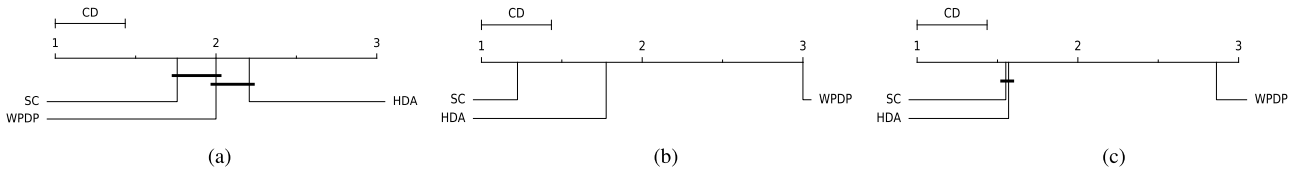


FIGURE 5. Comparison of HDA against WPDP and SC with Friedman test and Nemenyi's post-hoc test on the three indicators. (a) F-measure. (b) Balance. (c) AUC.

regarded as the most suitable performance indictor to evaluate the class imbalanced data, such as our software defect data. From this point of view, our method HDA and SC are both effective methods to conduct defect prediction on unlabeled defect data.

To sum up, our method HDA achieves competitive results compared with WPDP and a state-of-the-art unsupervised learning method on most specific cross-project pairs in terms of the three indicators.

VI. DISCUSSION

A. HOW MUCH TIME DOES IT TAKE FOR THE HCPDP METHODS TO PERFORM FEATURE TRANSFORMATION?

Besides effectiveness, we also evaluate the efficiency of HDA. Since the key of the three HCPDP methods in the experiment is to obtain the mapped project data (for HDA and CCA+) or the matched feature pairs (for HDP), we only consider the time used for the feature transformation or matching. When two projects take turns to act as the source project and target project, their time has a slight difference. For example, for the CCA+ method, the time used for feature transformation for the cross-project pair (cm1, H) (17.13 minutes) is very close to that for the cross-project pair (H, cm1) (17.07 minutes). Hence we take the average value 17.10 minutes as the time used for feature transformation for the two cross-project pairs based on cm1 and H. In addition, we find that the time for HDP and SC have small differences with the time for our HDA, hence Table 3 only reports the time (minutes) for HDA and CCA+.

TABLE 3. Time (minutes) for feature transformation of three HCPDP methods.

Pairs		HDA	HDP	CCA+	Pairs		HDA	HDP	CCA+
EQ	H	0.02	0.001	18.74	mw1	R	0.03	0.001	10.30
JDT	H	0.03	0.003	58.37	pc3	R	0.04	0.006	41.68
LC	H	0.03	0.002	39.93	pc4	R	0.05	0.007	58.09
ML	H	0.06	0.008	74.47	EQ	cm1	0.02	0.001	0.80
PDE	H	0.05	0.006	89.17	JDT	cm1	0.03	0.003	2.37
cm1	H	0.02	0.001	17.10	LC	cm1	0.03	0.002	1.66
mw1	H	0.02	0.001	13.91	ML	cm1	0.06	0.008	3.10
pc3	H	0.04	0.003	58.38	PDE	cm1	0.05	0.006	3.50
pc4	H	0.05	0.006	79.59	EQ	mw1	0.02	0.001	0.64
EQ	L	0.05	0.004	93.47	JDT	mw1	0.03	0.002	1.86
JDT	L	0.06	0.008	218.07	LC	mw1	0.03	0.001	1.29
LC	L	0.06	0.006	213.42	ML	mw1	0.06	0.007	2.50
ML	L	0.09	0.012	405.08	PDE	mw1	0.05	0.006	2.76
PDE	L	0.08	0.009	322.32	EQ	pc3	0.03	0.004	2.53
cm1	L	0.05	0.004	83.43	JDT	pc3	0.04	0.007	7.93
mw1	L	0.05	0.003	67.30	LC	pc3	0.04	0.006	5.48
pc3	L	0.06	0.008	274.77	ML	pc3	0.07	0.009	11.10
pc4	L	0.07	0.011	389.12	PDE	pc3	0.06	0.009	12.12
EQ	R	0.02	0.001	14.97	EQ	pc4	0.04	0.004	3.53
JDT	R	0.04	0.002	44.91	JDT	pc4	0.05	0.008	10.85
LC	R	0.03	0.001	31.06	LC	pc4	0.05	0.007	7.40
ML	R	0.06	0.007	56.94	ML	pc4	0.08	0.011	15.29
PDE	R	0.05	0.006	67.14	PDE	pc4	0.07	0.009	16.21
cm1	R	0.02	0.001	12.91	AVG		0.05	0.005	63.14

The figure shows that the time of HDP has significant difference than other two methods and the time of our method HDA has significant difference than CCA+. From the table, we observe that the time for CCA+ is at least 30 times longer than that for HDA. For example, for the cross-project pair based on pc3 and H, HDA runs about 1500 times faster than CCA+. The reason is that when the cross-project pairs containing projects with many entities (such as projects L, pc3, pc4, ML, PDE) or many features (such as the projects in the NetGene dataset), CCA+ runs slowly due to calculating the cross-project covariance matrix C_{ST} as mentioned in Section II-B. By contrast, the running time of heterogeneous feature mapping for HDA and HDP in all cross-project pairs is less than 0.1 minutes, which indicates that the efficiency of the two methods is not affected by the scale of the source project and target project. The reasons are that the process of feature selection and feature matching for HDP, and the LASSO solution, the partial derivation solution and the pseudo-inverse solution for HDA are all fast process in the reduced space. Since our datasets are collected from some large-scale software projects, it implies that HDA and HDP can be able to efficiently adapt to bigger software defect data for HCPDP.

B. HOW DIFFERENT CLASSIFIERS IMPACT THE PERFORMANCE OF HDA?

Although we select the logistic regression classifier as the basic prediction model, here, we investigate whether the transformed features by HDA perform well on other machine learning classifiers. To this end, we select the 7 classifiers as used in WPDP setting in Section V-B.

Fig. 6, 7, and 8 depict the boxplots of HDA with the seven classifiers on three HCPDP scenarios as well as all cross-project pairs scenario in terms of three indicators, respectively. In Fig. 6, HDA with LR classifier achieves the best median F-measure on all four scenarios. Besides LR, HDA with NB_K and CART usually perform better than with other classifiers in terms of median F-measure. In Fig. 7, the median Balance of HDA with LR classifier are much higher than with other classifiers on all four scenarios. In addition, HDA with NN_1 and CART can achieve better median Balance compared with other four classifiers. In Fig. 8, in terms of median AUC, HDA with LR classifier shows the superiority compared with other classifiers on all four scenarios. Moreover, HDA with NB_N and NN_8 can achieve satisfactory results on all scenarios.

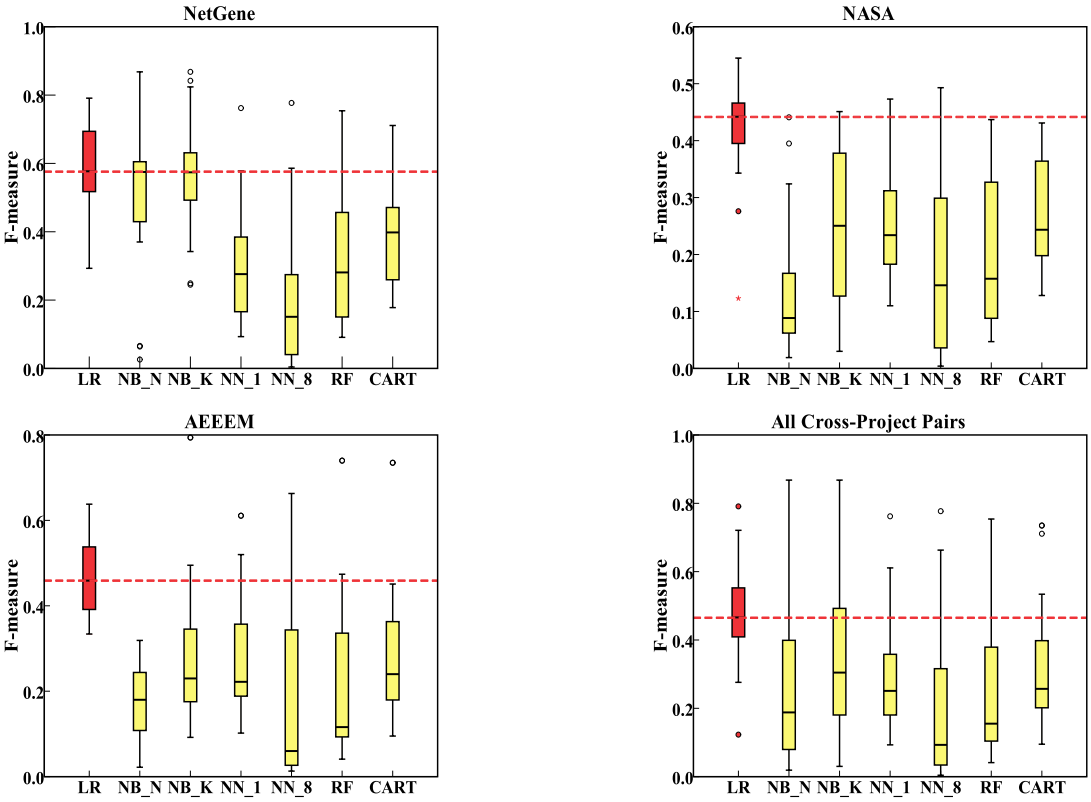


FIGURE 6. Boxplots of the F-measure values of HDA with seven classifiers.

From the above observations, we find that only LR can consistently achieve the best results in terms of the three indicators on all scenarios which is consistent with the conclusions in previous studies [66], [68], [69]. The reason is that HDA aims to find the optimal projective transformation to minimize the difference of the two mapped data as well as preserve the intrinsic similarity among the initial source and target project data. That is the reason that the LR classifier can obtain the better prediction performance. Whereas for other classifiers, the best classifier varies across different cross-project scenarios and indicators. Thus it is reasonable to select LR as our prediction model for HCPDP.

C. HOW DO DIFFERENT FEATURE DIMENSIONS SELECTED FOR THE MAPPED CROSS-PROJECT DATA IMPACT THE PERFORMANCE OF HDA?

HDA embeds the heterogeneous cross-project data into a lower-dimensional space. But it is non-trivial to determine the dimension of the feature space. Preserving too many features will introduce useless features that may decrease the prediction performances whereas selecting too fewer features will ignore important features that may contain favorable discrimination information. In this work, we choose five features for the mapped data as basic setting. Here, we explore how HDA performs with different feature dimensions. We empirically choose seven candidates: f_5, f_10, f_15, f_20, f_25, f_30, and f_35, no more than the features dimension (37) of the projects in NASA dataset.

Fig. 9 depicts the boxplots of HDA with the seven candidates across all cross-project pairs in terms of the three indicators. For the median indicator values, HDA achieves favorable results with the feature dimensions of 5 and 10. In addition, the performance of HDA decreases as the feature dimension increases. Thus, small parameter values are the optimal configuration for feature dimension in HDA. In this paper, we choose the dimension as 5 to conduct all our experiments.

D. HOW DO THE PARAMETER γ AFFECT THE PERFORMANCE OF HDA?

Parameter γ is used to control the sparsity of the coefficient matrices during the dictionary representation process. We select $\gamma = 0.5$ in our experiment. Here, we evaluate the influence of different γ values on the experimental results by selecting ten parameter values (i.e., C_0.1, C_0.2, ..., C_1.0) for comparison.

Fig. 10 depicts the boxplots of HDA with the ten γ values across all cross-project pairs in terms of the three indicators. From the figure, we observe that HDA with different γ values achieve almost the same median F-measure and Balance. In addition, HDA with $\gamma = 0.5$ achieves slightly higher median AUC compared with other nine γ values, but the

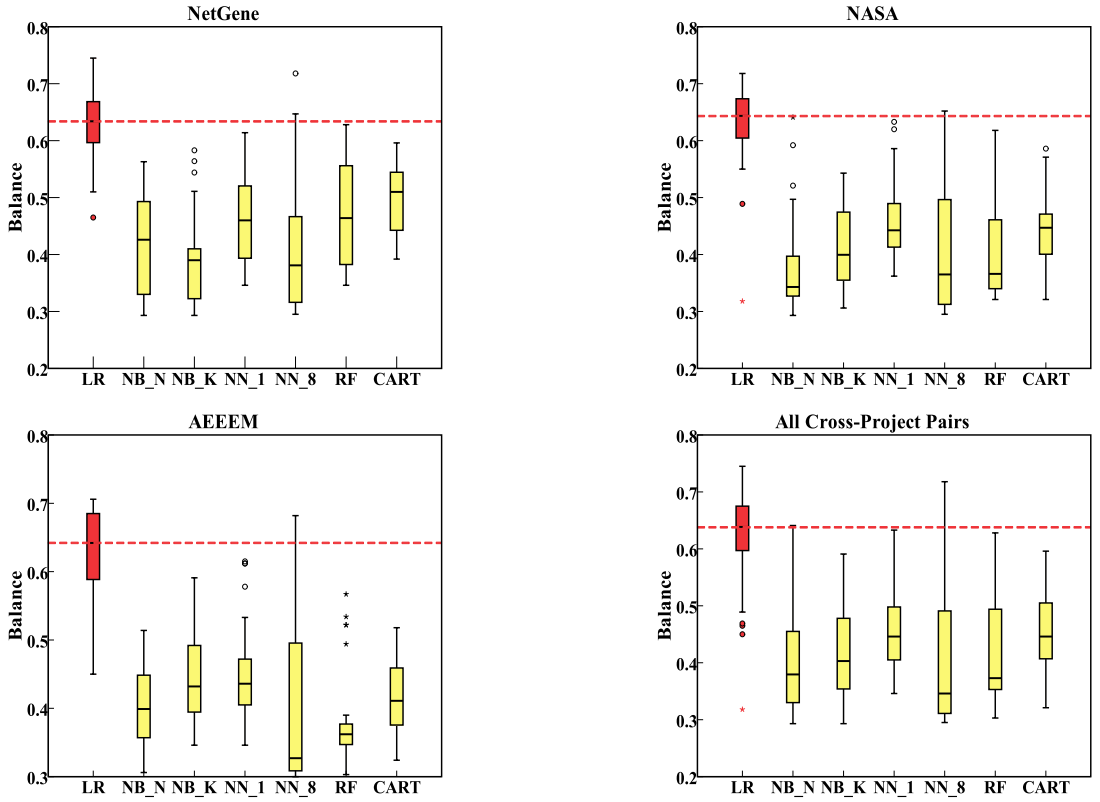


FIGURE 7. Boxplots of the Balance values of HDA with seven classifiers.

differences are very small. So this parameter has litter impact on the performance of HDA for HCPDP.

E. HOW THE PARAMETER λ IMPACT THE PERFORMANCE OF HDA?

Parameter λ is used to control the difference degree between the two learned dictionaries. we set $\lambda = 5$ in our experiment without any prior knowledge. Here we evaluate whether different λ values impact the performances of HDA by selecting twelve values (i.e., L_5, L_10, L_20, L_40, …, L_200) for comparison.

Fig. 11 depicts the boxplots of HDA with the twelve λ values across all cross-project pairs in terms of three indicators, respectively. we observe that the median values of the three indicators decrease as the parameter value increases, and HDA with $\lambda = 5$ obtains relatively higher median values for all three indicators. In general, small λ values are more effective for HDA to achieve better HCPDP performance.

F. HOW THE PARAMETER η IMPACT THE PERFORMANCE OF HDA?

Parameter η is used to control the stability of the solution of our HDA method to avoid the two mapped matrics too large. We set $\eta = 0.001$ in our experiment. Here we investigate whether different η values impact the performances of HDA by selecting ten values (i.e., E_0.001, E_0.002, …, E_0.01) for comparison.

Fig. 12 depicts the boxplots of HDA with the ten η values across all cross-project pairs in terms of three indicators, respectively. From the figure, we observe that HDA with different η values achieve almost the same median F-measure and Balance, while HDA with $\eta = 0.001$ achieves higher median AUC compared with other nine options. Overall, setting $\eta = 0.001$ is suitable for our HDA method.

VII. THREATS TO VALIDITY

In this section, we discuss the threats to the validity of this work.

A. THREATS TO CONSTRUCT VALIDITY

Construct validity focuses on the suitability of the used performance indicators. In this work, we choose three commonly-used indicators as our measurement criteria, which enables us to have a comprehensive evaluation of our method compared with the baseline methods.

B. THREATS TO INTERNAL VALIDITY

Internal validity relates to the impact of method implementation, parameter setting on the conclusion. Since the

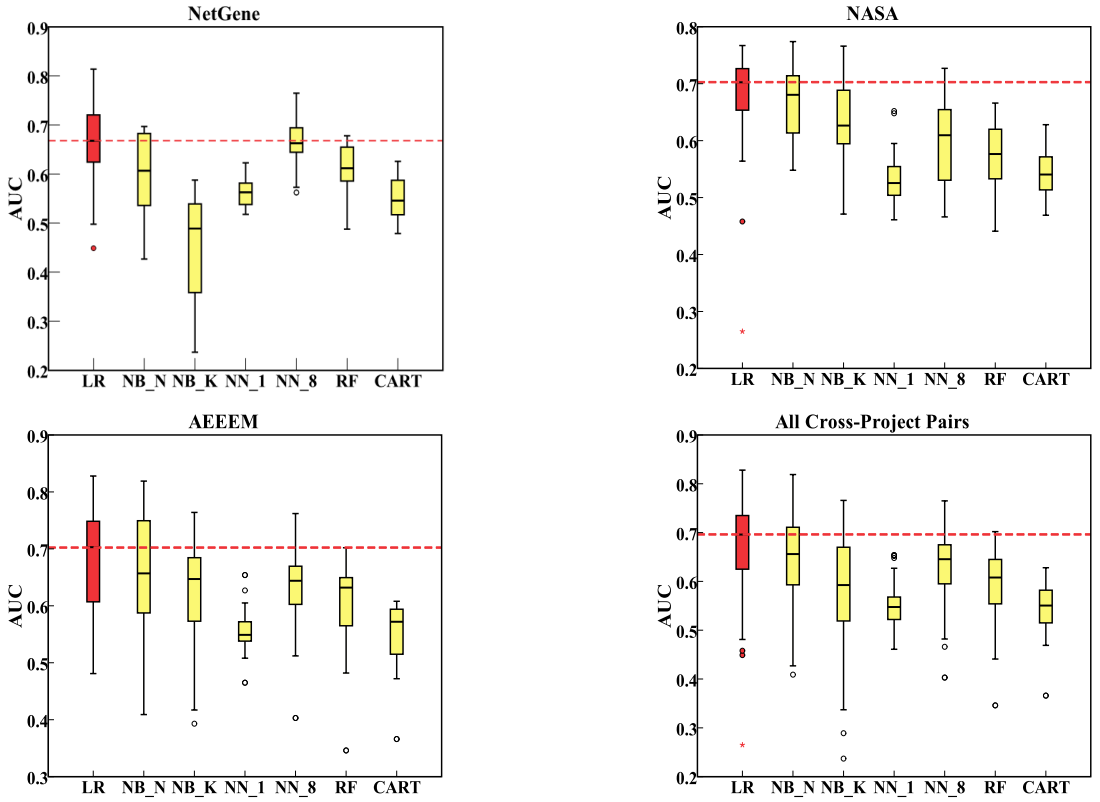


FIGURE 8. Boxplots of the AUC values of HDA with seven classifiers.

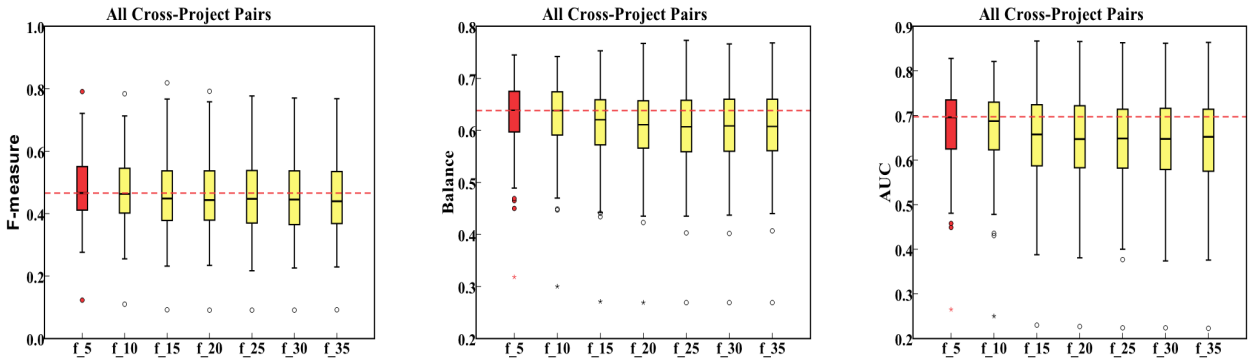


FIGURE 9. Boxplots of the three indicator values of HDA with different feature dimensions on all cross-project scenario.

authors do not make the source codes of their methods available, we carefully implement the two state-of-the-art HCPDP methods following the algorithm descriptions in corresponding papers. In addition, for the CCA+ method, the authors do not state feature dimensions of the two mapped data. For a fair comparison, we choose five features for CCA+ as selected for HDA, thus the results may be incompletely consistent with that in [16]. For the parameter setting, we tune them by searching the optimal values of one parameter from a given range after fixing the other parameters. A more rigorous grid-based search method is needed to seek the optimal parameter combination.

C. THREATS TO EXTERNAL VALIDITY

External validity pays attention to the generalization of the results of our study to other software defect datasets. We alleviate the bias by carefully choosing three public open-source defect datasets whose features are collected at different levels of granularity as our benchmark datasets. Validating the effectiveness and efficiency of HDA on more

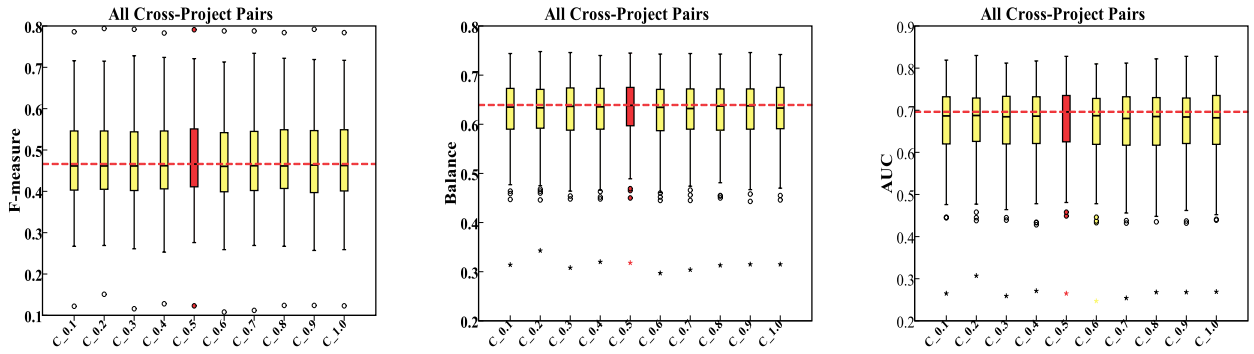


FIGURE 10. Boxplots of the three indicator values of HDA with different γ values on all cross-project scenario.

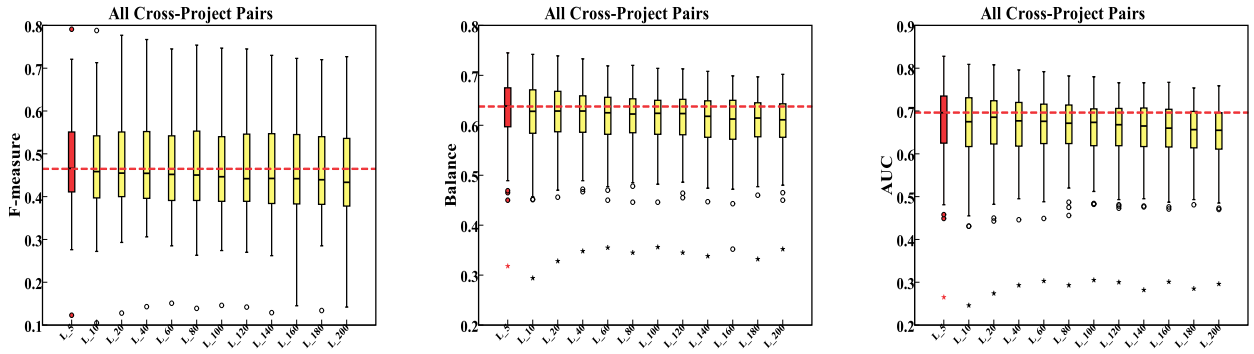


FIGURE 11. Boxplots of the three indicator values of HDA with different λ values on all cross-project scenario.

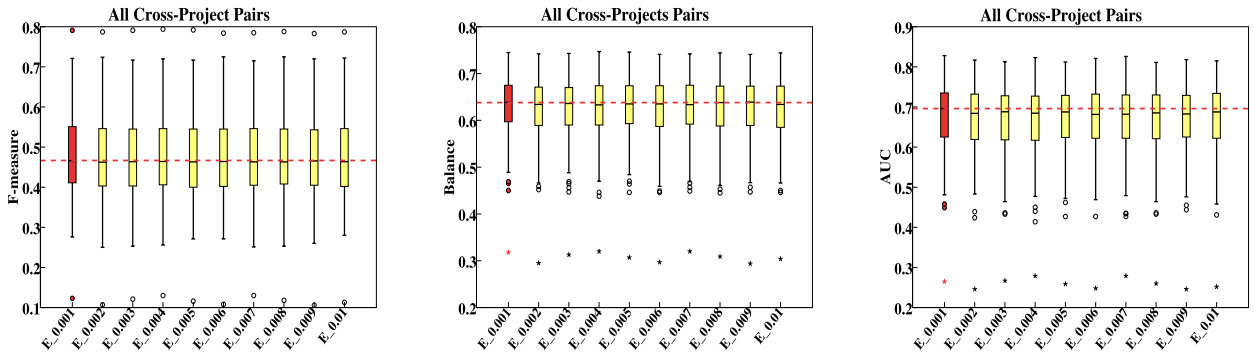


FIGURE 12. Boxplots of the three indicator values of HDA with different η values on all cross-project scenario.

diversity and bigger datasets with heterogeneous features is needed.

VIII. CONCLUSION

It is challenging to conduct defect prediction on heterogeneous cross-project data since the features among them have no correlation due to different meanings and distributions. We propose a novel method HDA to address this problem by treating the cross-project data as being from two heterogeneous domains. HDA maps the two domains of data into a common space and measures the difference between the mapped data with the dictionaries obtained from them by a dictionary learning technique. Different from existing HCPDP methods that only work on some cross-project pairs, HDA can be applied to any pair. Extensive experiments indicate that HDA is superior to the two advanced HCPDP methods in most cases in terms of three indicators, and HDA can achieve competitive performances compared with WPDP scenario and a state-of-the-art unsupervised learning method in many cross-project pairs.

As software defect datasets (such as NASA and AEEEM datasets) usually encounter class imbalance issue, i.e., defective entities are much fewer than non-defective entities, it will threaten the defect prediction performance. However, we do not consider this issue here since this paper mainly focuses on dealing with heterogeneous feature mapping issue for two defect datasets with different features. We will enhance our HDA method to further improve HCPDP performance by taking the class imbalance into consideration in future work. In addition, since our method HDA exists randomness due to randomly selecting the entities to initialize the dictionary atoms in k-SVD algorithm, we plan to optimize HDA and eliminate the randomness by employing the clustering methods to select the most important entities as the initial atoms.

ACKNOWLEDGMENT

(Zhou Xu and Peipei Yuan contributed equally to this work.)

REFERENCES

[1] M. M. Ali, S. Huda, J. Abawajy, S. Alyahya, H. Al-Dossari, and J. Yearwood, "A parallel framework for software defect detection and metric selection on cloud computing," *Cluster Comput.*, vol. 20, no. 3, pp. 2267–2281, 2017.

[2] J. C. Munson and T. M. Khoshgoftaar, "The detection of fault-prone programs," *IEEE Trans. Softw. Eng.*, vol. 18, no. 5, pp. 423–433, May 1992.

[3] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Trans. Softw. Eng.*, vol. 31, no. 10, pp. 897–910, Oct. 2005.

[4] P. L. Li, J. Herbsleb, M. Shaw, and B. Robinson, "Experiences and results from initiating field defect prediction and product test prioritization efforts at ABB Inc," in *Proc. 28th Int. Conf. Softw. Eng.*, 2006, pp. 413–422.

[5] A. E. Hassan, "Predicting faults using the complexity of code changes," in *Proc. IEEE 31st Int. Conf. Softw. Eng. (ICSE)*, 2009, pp. 78–88.

[6] X. Xia, D. Lo, S. J. Pan, N. Nagappan, and X. Wang, "HYDRA: Massively compositional model for cross-project defect prediction," *IEEE Trans. Softw. Eng.*, vol. 42, no. 10, pp. 977–998, Oct. 2016.

[7] J. Xuan *et al.*, "Towards effective bug triage with software data reduction techniques," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 264–280, Jan. 2015.

[8] N. Fenton *et al.*, "Predicting software defects in varying development lifecycles using Bayesian nets," *Inf. Softw. Technol.*, vol. 49, no. 1, pp. 32–43, 2007.

[9] S. Kanmani, V. R. Uthariaraj, V. Sankaranarayanan, and P. Thambidurai, "Object-oriented software fault prediction using neural networks," *Inf. Softw. Technol.*, vol. 49, no. 5, pp. 483–492, 2007.

[10] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–13, Jan. 2007.

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

[12] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proc. Int. Conf. Softw. Eng.* Piscataway, NJ, USA: IEEE Press, May 2013, pp. 382–391.

[13] L. C. Briand, W. L. Melo, and J. Wust, "Assessing the applicability of fault-proneness models across object-oriented software projects," *IEEE Trans. Softw. Eng.*, vol. 28, no. 7, pp. 706–720, Jul. 2002.

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

[15] B. Turhan, "On the dataset shift problem in software engineering prediction models," *Empirical Softw. Eng.*, vol. 17, nos. 1–2, pp. 62–74, 2012.

[16] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, "Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning," in *Proc. 10th Joint Meeting Found. Softw. Eng.*, 2015, pp. 496–507.

[17] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous defect prediction," *IEEE Trans. Softw. Eng.*, vol. 44, no. 9, pp. 874–896, Sep. 2017.

[18] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process," in *Proc. 7th Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng.*, 2009, pp. 91–100.

[19] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Negative samples reduction in cross-company software defects prediction," *Inf. Softw. Technol.*, vol. 62, pp. 67–77, Jun. 2015.

[20] S. Kong and D. Wang, "Transfer heterogeneous unlabeled data for unsupervised clustering," in *Proc. 21st Int. Conf. Pattern Recognit. (ICPR)*, Nov. 2012, pp. 1193–1196.

[21] C. Persello and L. Bruzzone, "Active learning for domain adaptation in the supervised classification of remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 11, pp. 4468–4483, Nov. 2012.

[22] C. Persello and L. Bruzzone, "A novel active learning strategy for domain adaptation in the classification of remote sensing images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2011, pp. 3720–3723.

[23] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 2960–2967.

[24] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Nov. 2011, pp. 999–1006.

[25] L. Duan, D. Xu, and I. Tsang. (2012). "Learning with augmented features for heterogeneous domain adaptation." [Online]. Available: https://arxiv.org/abs/1206.4660

[26] Q. Yang, Y. Chen, G.-R. Xue, W. Dai, and Y. Yu, "Heterogeneous transfer learning for image clustering via the social Web," in *Proc. Joint Conf. 47th Annu. Meeting ACL 4th Int. Joint Conf. Natural Lang. Process. (AFNLP)*, vol. 1, 2009, pp. 1–9.

[27] X. Shi, Q. Liu, W. Fan, and P. S. Yu, "Transfer across completely different feature spaces via spectral embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 906–918, Apr. 2013.

[28] M. Harel and S. Mannor. (2010). "Learning from multiple outlooks." [Online]. Available: https://arxiv.org/abs/1005.0027

[29] F. Zhang, I. Keivanloo, and Y. Zou, "Data transformation in cross-project defect prediction," *Empirical Softw. Eng.*, vol. 22, no. 6, pp. 3186–3218, 2017.

[30] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[31] Z. Wang, Y. Song, and C. Zhang, "Transferred dimensionality reduction," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Berlin, Germany: Springer, 2008, pp. 550–565.

[32] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 801–808.

[33] D. S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas With Application to Linear Systems Theory*, vol. 41. Princeton, NJ, USA: Princeton Univ. Press, 2005.

[34] K. Herzig, S. Just, A. Rau, and A. Zeller, "Predicting defects using change genealogies," in *Proc. IEEE 24th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Nov. 2013, pp. 118–127.

[35] M. D'Ambros, M. Lanza, and R. Robbes, "An extensive comparison of bug prediction approaches," in *Proc. 7th IEEE Work. Conf. Mining Softw. Repositories (MSR)*, May 2010, pp. 31–41.

[36] J. Nam and S. Kim, "CLAMI: Defect prediction on unlabeled datasets (T)," in *Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2015, pp. 452–463.

[37] Z. Xu, J. Xuan, J. Liu, and X. Cui, "MICHAC: Defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering," in *Proc. IEEE 23rd Int. Conf. Softw. Anal., Evol., Reeng. (SANER)*, vol. 1, Mar. 2016, pp. 370–381.

[38] K. Herzig, S. Just, A. Rau, and A. Zeller, "Classifying code changes and predicting defects using changegenealogies," Saarland Univ., Saarbrücken, Germany, Tech. Rep., 2013.

[39] N. Pingclasai, H. Hata, and K.-I. Matsumoto, "Classifying bug reports to bugs and other requests using topic modeling," in *Proc. 20th Asia–Pacific Softw. Eng. Conf. (APSEC)*, vol. 2, Dec. 2013, pp. 13–18.

[40] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, A. Ihara, and K. Matsumoto, "The impact of mislabelling on the performance and interpretation of defect prediction models," in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng. (ICSE)*, vol. 1, May 2015, pp. 812–823.

[41] S. Watanabe, H. Kaiya, and K. Kaijiri, "Adapting a fault prediction model to allow inter languagereuse," in *Proc. 4th Int. Workshop Predictor Models Softw. Eng.*, 2008, pp. 19–24.

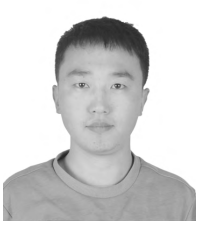
[42] Y. Jiang, B. Cukic, and Y. Ma, "Techniques for evaluating fault prediction models," *Empirical Softw. Eng.*, vol. 13, no. 5, pp. 561–595, 2008.
[43] B. Turhan and A. Bener, "A multivariate analysis of static code attributes for defect prediction," in *Proc. 7th Int. Conf. Qual. Softw. (QSIC)*, Oct. 2007, pp. 231–237.
[44] Q. Song, Z. Jia, M. J. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, pp. 356–370, May 2011.
[45] M. Li, H. Zhang, R. Wu, and Z.-H. Zhou, "Sample-based software defect prediction with active and semi-supervised learning," *Automated Softw. Eng.*, vol. 19, no. 2, pp. 201–230, 2012.
[46] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.
[47] D. W. Hosmer, Jr., S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, vol. 398. Hoboken, NJ, USA: Wiley, 2013.
[48] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 485–496, Jul. 2008.
[49] X. Yang, D. Lo, X. Xia, Y. Zhang, and J. Sun, "Deep learning for just-in-time defect prediction," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Aug. 2015, pp. 17–26.
[50] Y. Yang *et al.*, "Are slice-based cohesion metrics actually useful in effort-aware post-release fault-proneness prediction? An empirical study," *IEEE Trans. Softw. Eng.*, vol. 41, no. 4, pp. 331–357, Apr. 2015.
[51] Y. Yang *et al.*, "An empirical study on dependence clusters for effort-aware fault-proneness prediction," in *Proc. 31st IEEE/ACM Int. Conf. Automated Softw. Eng.*, 2016, pp. 296–307.
[52] A. Mockus, P. Zhang, and P. L. Li, "Predictors of customer perceived software quality," in *Proc. 27th Int. Conf. Softw. Eng.*, May 2005, pp. 225–233.
[53] A. Panichella, R. Oliveto, and A. De Lucia, "Cross-project defect prediction models: L'Union fait la force," in *Proc. Softw. Evol. Week-IEEE Conf. Softw. Maintenance, Reeng. Reverse Eng. (CSMR-WCRE)*, Feb. 2014, pp. 164–173.
[54] Y. Liu, T. M. Khoshgoftaar, and N. Seliya, "Evolutionary optimization of software quality modeling with multiple repositories," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 852–864, Nov. 2010.
[55] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Trans. Softw. Eng.*, vol. 22, no. 10, pp. 751–761, Oct. 1996.
[56] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," in *Proc. 28th Int. Conf. Softw. Eng.*, 2006, pp. 452–461.
[57] F. Rahman, D. Posnett, and P. Devanbu, "Recalling the 'imprecision' of cross-project defect prediction," in *Proc. ACM SIGSOFT 20th Int. Symp. Found. Softw. Eng.*, 2012, p. 61.
[58] S. Zhong, T. M. Khoshgoftaar, and N. Seliya, "Unsupervised learning for expert-based software quality estimation," in *Proc. HASE*, Mar. 2004, pp. 149–155.
[59] F. Zhang, Q. Zheng, Y. Zou, and A. E. Hassan, "Cross-project defect prediction using a connectivity-based unsupervised classifier," in *Proc. 38th Int. Conf. Softw. Eng.*, 2016, pp. 309–320.
[60] T. Wang, Z. Zhang, X. Jing, and L. Zhang, "Multiple kernel ensemble learning for software defect prediction," *Automated Softw. Eng.*, vol. 23, no. 4, pp. 569–590, 2016.
[61] X.-Y. Jing, S. Ying, Z.-W. Zhang, S.-S. Wu, and J. Liu, "Dictionary learning based software defect prediction," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 414–423.
[62] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *Proc. 37th Int. Conf. Softw. Eng.*, vol. 1. Piscataway, NJ, USA: IEEE Press, 2015, pp. 789–800.
[63] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.
[64] T. Mende and R. Koschke, "Effort-aware defect prediction models," in *Proc. 14th Eur. Conf. Softw. Maintenance Reeng. (CSMR)*, Mar. 2010, pp. 107–116.
[65] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: A benchmark and an extensive comparison," *Empirical Softw. Eng.*, vol. 17, nos. 4–5, pp. 531–577, 2012.
[66] Z. Li, X.-Y. Jing, X. Zhu, H. Zhang, B. Xu, and S. Ying, "On the multiple sources and privacy preservation issues for heterogeneous defect prediction," *IEEE Trans. Softw. Eng.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/8168387
[67] S. Herbold, A. Trautsch, and J. Grabowski, "A comparative study to benchmark cross-project defect prediction approaches," *IEEE Trans. Softw. Eng.*, vol. 44, no. 9, pp. 811–833, Sep. 2017.
[68] Z. Li, X.-Y. Jing, F. Wu, X. Zhu, B. Xu, and S. Ying, "Cost-sensitive transfer kernel canonical correlation analysis for heterogeneous defect prediction," *Automated Softw. Eng.*, vol. 25, no. 2, pp. 201–245, 2018.
[69] Z. Li, X.-Y. Jing, and X. Zhu, "Heterogeneous fault prediction with cost-sensitive domain adaptation," *Softw. Test., Verification Rel.*, vol. 28, no. 2, p. e1658, 2018.

ZHOU XU is currently pursuing the joint Ph.D. degree with the School of Computer Science, Wuhan University, China, and also with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He is also a temporary full-time Research Assistant with the College of Computer Science and Technology, Harbin Engineering University, China. His research interests include software defect prediction, feature engineering, and data mining.

PEIPEI YUAN received the B.Sc. degree in information and computation science from Huazhong Agricultural University in 2014. She is currently pursuing the Ph.D. degree with the School of Electronic Information and Communications, Huazhong University of Science and Technology, China. Her research interests include computer vision, pattern recognition, machine learning, and data mining.

TAO ZHANG received the Ph.D. degree in computer science from the University of Seoul, South Korea, and the master's and bachelor's degree at Northeastern University, China. He is currently an Associate Professor with the College of Computer Science and Technology, Harbin Engineering University. He spent a year as a Post-Doctoral Research Fellow with the Department of Computing, The Hong Kong Polytechnic University. His research interests include mining software repositories and empirical software engineering.

YUTIAN TANG received the Ph.D. degree from The Hong Kong Polytechnic University in 2018, and the B.Sc. degree from Jilin University in 2013. He is currently a Research Staff with The Hong Kong Polytechnic University. His research interests include software product line, system security, and machine learning.



SHUAI LI received the B.S. degree in computer science from the Dalian University of Technology in 2017. He is currently pursuing the Ph.D. degree with the Department of Computing, The Hong Kong Polytechnic University. His research interests are in classification and detection both in theory and applications.

ZHEN XIA received the B.Sc. degree from the School of Computer Science, Wuhan University, in 2018. His research interests include feature engineering and machine learning.