LPDA: Cross-Project Software Defect Prediction Approach via Locality Preserving and Distribution Alignment

Jin Xian[1], Jinglei Li[2]*, Quanyi Zou[3], Yunting Xian[4]
School of Computer Science and Engineering South China University of Technology Guangzhou 510006, China[1,4]
The Second Branch of China Railway Electrification Engineering Bureau Group Co., Ltd Guangzhou 511492, China[2]
School of Journalism and Communication South China University of Technology Guangzhou 510006, China[3]
Guangzhou 510006, China Guangdong Yousuan Technology Co., Ltd, FoShan 528000, P.R.China[4]

***Abstract*—Cross-Project Defect Prediction (CPDP) based on domain adaptation aims to achieve defect prediction tasks in an unlabeled target software project by borrowing the defect knowledge extracted from well-annotated source software projects. Most existing CPDP approaches enhance transferability between projects but struggle with misalignments due to limited exploration of class-specific features and inability to preserve original local relationships in transformed features. In order to tackle these challenges, The article introduces a novel Cross-Project Defect Prediction (CPDP) approach called Local Preserving and Distribution Alignment (LPDA). This approach addresses the challenge of misalignments in CPDP due to limited exploration of discriminative feature representations and the failure to preserve original local relationship consistency. LPDA combines transferability and discriminability for CPDP tasks. It uses locality-preserving projection to maintain module consistency and distribution alignment, which includes transferable and discriminant distribution alignment. The former narrows the distributions of both source and target projects, while the latter increases the discrepancy between different classes across projects. The effectiveness of LPDA was tested through 118 cross-project prediction tasks involving 22 software projects from four distinct repositories. The results showed that LPDA outperforms baseline CPDP methods by efficiently learning representations that integrate transferability and discriminability while preserving local geometry to optimize distances within and between categories.**

***Keywords*—Cross project defect prediction; discriminative distribution alignment; local preserving; domain adaption**

I. Introduction

As software grows in size and complexity, defects inevitably arise, compromising quality and security [1], [2]. Ensuring software quality is therefore crucial before its release [3]. Software Defect Prediction (SDP) is a key technique to improve reliability by identifying potential defects in software modules, allowing for better allocation of testing resources. This technique uses historical data, like past source code and defect reports, to build models that can predict defects in new modules. Common methods for creating these models include neural networks [4], Naive Bayes [5], and support vector machines [6]. When prediction models are based on data from the same project, it's known as Within Project Defect Prediction (WPDP) [7], [8]. However, not all companies maintain historical defect data, and for those scenarios, Cross Project Defect Prediction (CPDP) uses data from external projects to build prediction models [9], [10].

CPDP aims to achieve defect prediction tasks in an unlabeled target project by learning the defect knowledge obtained from a source software project [11], [12]. However, Defect prediction for a target project is challenging due to differences from the source project, such as coding languages and developer expertise, which prevent direct knowledge transfer [13]. Domain adaptation is used to bridge the gap between projects, allowing defect knowledge to be transferred by adjusting features or instances in CPDP methods [14]. Various transfer learning algorithms from the literature [15], [16], [17], [18] are used to narrow marginal and/or conditional distribution difference between two projects. The CPDP approaches based on instance level select or reweight appropriate instances to decline the unfortunate impact from irrelevant cross-project data. For example, Ma et al. [19] proposed the Transfer Naive Bayes (TNB), introducing data gravitation that reweights instances of the source project. Moreover, software defect data often exhibits class-imbalance, with a significantly larger number of non-defective instances compared to defective instances. Class-imbalance has been broadly investigated both in WPDP [20], [21] and CPDP [22], [23], [24]. The impact of imbalanced datasets on the ranking of the approaches is also assessed [25], which addresses both class-imbalance and distribution mismatching. Tong et al. [11] introduced KSETE (Kernel Spectral Embedding Transfer Ensemble) to tackle class-imbalance in both homogeneous cross-project and heterogeneous cross-project scenarios.

Existing CPDP methods effectively reduce the gap between source and target projects, yet they overlook class distinction and disrupt local instance relationships. This can blur the decision boundary and misplace instances in the feature space, making accurate predictions difficult. Integrating locality-preserving techniques with domain adaptation could improve performance by maintaining the original data structure. Locality preserving projection is a typical approach based on manifold learning [26] that allows for learning a favourable feature space where the local consistency in the raw feature space can be effectively retained. The perfect performance will be obtained by integrating locality preserving projection and domain adaptation [27], [28], [29], but Locality-preserving objectives have not received thorough exploration within the CPDP domain.

To overcome the aforementioned limitations, this article introduces a new CPDP method called Local Preserving and Distribution Alignment (LPDA), which focuses on maintaining class distinctions and local data relationships. It aligns distributions globally and locally while keeping instances of the same class close and separating different classes. Additionally, new representations are generated in a low-dimensional subspace where instances instances from the same class remain closely, while instances from different classes are positioned farther apart. Extensive CPDP tasks on 22 open-source projects from four software repositories validate the effectiveness of the proposed approaches. The evaluation metrics used included F-measure, Balance, MCC, and AUC. The Wilcoxon signed rank test and Scott-Knott ESD test were adopted to statistical significance test. The contributions of this article can be summarised as follows:

1) Our proposed CPDP method, unlike previous ones, enhances transferability by reducing distribution discrepancies and also focuses on discriminability between different classes in the projects..
2) In CPDP, using locality-preserving projection maintains local consistency within classes, keeping similar instances closer and distinctly separating different categories.
3) The extensive experiments on 22 software projects from four software repositories demonstrate that the proposed LPDA approach is superior to several state-of-the-art CPDP approaches in terms of four performance indicators.

The article is organized as follows. Section II reviews prior works on CPDP and subspace alignment-based domain adaptation. Section III introduces the technical specification of our LPDA approach. Section IV covers the experiment configuration, encompassing aspects like open datasets, statistical tests, evaluation criteria, and research questions. In Section V, Extensive experiments along with analysis under CPDP scenarios are presented in detail.

II. Related Work

In this section, we will provide a concise overview of previous research in the domains of cross-project defect prediction and domain adaptation based on feature alignment.

A. Cross Project Defect Prediction

CPDP aims to seek instances with a high likelihood of defects in an unlabeled target software project using a predictive model trained on other well-labeled software projects[30]. In earlier research, Zimmermann et al. [31] investigated different factors that might influence the cross-project prediction performance for the first time. They have found that a few tasks (only 3.4% of the cross-project tasks) could achieve adequate prediction results and CPDP tasks between the projects are not symmetrical. The current CPDP approaches can be broadly categorized into two groups: homogeneous cross-project and heterogeneous cross-project, based on the similarity of software metrics (features) [11], [32]. In the context of homogeneous cross-project, the source and target projects share the same feature space, whereas in heterogeneous cross-project, the features of source and target projects differ. The CPDP approach employed in this paper falls under the category of homogeneous cross-project.

According to the theory of knowledge transfer, the current CPDP approaches can be mainly divided into instance transferring and feature transferring. The CPDP approaches based on instance transferring seek to select or reweight relevant instances from the source project data, which can be advantageous for the target task. The CPDP approaches based on feature transferring pay attention to learning shared feature representations for the two projects so as to narrow the distribution discrepancy between them. The former either chooses or adjusts the training data to mitigate the influence of detrimental information. The latter ensures that the source and the target projects exhibit a comparable distribution within the newly created feature subspace.

To the best of our knowledge, Turhan et al. [33] presented the CPDP approach that focus on instance transferring named Nearest Neighbor filter (NN-filter). They used KNN to gather close instances together to construct a similar training dataset.

In greater detail, for every instance of target project, NN-filter method picks the ten nearest instances from the source project and subsequently incorporates them into the training dataset. Building upon the NN-filter, Peters et al. [34] presented the Peter-filter approach, which selected instances using the k-means clustering algorithm. Using different clustering algorithms to select instances, Kanwata et al. [35] and Bhat et al. [36] presented two different CPDP approaches. Lately, Hosseini et al. [37] presented a search-based genetic instance selection approach, using genetic algorithm to select training data. These CPDP approaches based on instance selecting led to the source project wasting some data or a few available instances. To solve these problems, Ma et al. [19] proposed TNB, using the concept of data gravity, the transfer weights from the source projects is computed. These weights could strengthen the instances with significant correlations and weaken the impact of ineffective instances.

Drawn from transfers component analysis (TCA) [38], a classical transferring learning algorithm, Nam et al. [17] proposed the TCA+. This method adds normalization rules to process source and target data before distribution alignment. Liu et al. [39] detected that the prediction performance of TCA+ is erratic. Thus, they proposed two-phase transfer learning (TPTL) approach to m the issue of instability. TCA+ and TPTL only narrow the disparity in marginal distribution between the source and target projects. Simultaneously considering both the marginal and conditional distributions, Qiu et al.[15] and Xu et al. [18] proposed joint distribution matching (JDM) and balanced distribution adaptation (BDA), respectively.

Since deep learning has become capable of automatically extracting semantic features from ASTs of software program, many researchers have used it in the research SDP. Wang et al. [40] claimed that only used traditional software metrics were far from enough and represented the relationship between semantic features and software programs by abstract syntax tree (AST). Then they applied a deep belief network (DBN) to obtain software code semantic features from ASTs. Subsequently, Li et al. [41] constructed a extracting feature approach based on CNN to extract semantic features, and then integrated this features and traditional software metrics to

address the absence of semantic information. However, these shallow learning and deep learning CPDP approaches consider only the transferability between source and target projects but ignore discriminability between classes.

B. Feature Alignment-based Domain Adaptation

Feature alignment-based domain adaptation approaches aim to learn a low-dimensional feature subspace where the disparity in distribution between the source and target data is explicitly narrowed. As a pioneer, Pan et al. [38] proposed to map both source and target data into a shared feature subspace via TCA, which aligns the marginal distributions between the source and target domains while maximizing the data variance in adaptation process. However, focusing only on aligning marginal distributions is insufficient for better learning purposes. Therefore, Long et al. [42] proposed to align both marginal and conditional distributions between two domains via joint distribution analysis (JDA). Similarly, Wang et al. [43] proposed a balanced domain adaptation (BDA), which simultaneously aligns the marginal and conditional distribution discrepancy and exploits a balance factor to adjust their importance degrees. JDA and BDA employ the classifier trained on source domain data to generate pseudo labels for the unlabeled target domain. After that, many variants will emerge. For example, Wang et al. [44] further proposed manifold embedded distribution alignment (MEDA) to dynamically adjust the relative importance of these two distributions. Zhao et al. [45] proposed discriminative joint probability MMD (DJP-MMD), which not only minimizes the divergence in joint distribution between two domains, but also maximizes the divergence in joint probability distribution between distinct classes in different domains to learn discriminative feature representation.

III. Research Methodology

In this section, we will present the LPDA approach for CPDP in more details. After describing the descriptions of notations used in this article.

A. Notations

$\mathcal{D}_s = \{\mathbf{x}_i^s, y_i^s\}_{i=1}^{n_s}$ and $\mathcal{D}_t = \{\mathbf{x}_j^t\}_{j=1}^{n_t}$ respectively represent the source project and the target project under the assumptions that both marginal distributions and conditional distributions of source and target projects are inequality($P(\mathbf{x}^s) \neq P(\mathbf{x}^t)$ and $Q(y^s \mid \mathbf{x}^s) \neq Q(y^t \mid \mathbf{x}^t)$). Let $\mathbf{X}_s \in \mathbb{R}^{d \times n_s}$ ($\mathbf{X}_t \in \mathbb{R}^{d \times n_t}$) is the source project data matrix (target project data matrix) containing $n_s(n_t)$ instance with d-dimension. $\mathbf{X}^c$ denotes a set of instances with label c. $\mathbf{x}_i^s \in \mathbf{X}_s$ and $\mathbf{x}_j^t \in \mathbf{X}_t$ are the $i-th$, $j-th$ instances in the source project and target project, respectively. The proposed CPDP approach aims to learn a transformation matrix $\mathbf{A}$ to transform the instances from the original feature space into a low-dimensional subspace where the distributions can be aligned and the important properties of the original data can be preserved. M_{mar} and M_{con} denote the distribution matching of marginal and the conditional distributions, respectively.

B. Overall Framework of LPDA

The previous CPDP approaches, such as JDM [15], TCA+ [17], TPTL [39] and BDA [18], learn global feature representation to narrow the distribution gap between different projects for the purpose of higher transferability, but disregard the discriminability between different classes. Additionally, these CPDP approaches cannot well preserve original local relationship consistency instances of shared the same label after transforming the features. To address these problems, this article proposes a novel CPDP approaches called Locality Preserving and Distribution Alignment (LPDA). Peculiarly, LPDA tries to fulfill three complementary objectives as follows. (1)

1) It aims at the characteristics of cross-project significant variations and draws from the idea of domain adaptation to reduce the global inter-project difference and the local intra-class discrepancy for to increaser transferability.
2) For discriminability between defective and non-defective classes, LPDA leverages class-wise MMD to enlarge the distance of different classes.
3) LPDA introduces the reward graph and penalty graph to maximize preserve the geometric structure of project instances.

The overall objective function of LPDA is follows:

$$\begin{aligned}\underset{\mathbf{A}}{\mathcal{L}} = \arg\min_{\mathbf{A}} & \underbrace{\mathcal{M}_t(\mathbf{X}_s, \mathbf{X}_t, \mathbf{A})}_{\text{transferability}} - \mu \underbrace{\mathcal{M}_d(\mathbf{X}_s, \mathbf{X}_t, \mathbf{A})}_{\text{discriminability}} \\ & + \eta \underbrace{\mathcal{G}(\mathbf{X}_s, \mathbf{X}_t, \mathbf{A})}_{\text{geometric structure}} + \lambda \underbrace{\Omega(\mathbf{A})}_{\text{regularization}}. \end{aligned} \tag{1}$$

In Eq. (1), $\mathcal{M}_t$ is distribution difference in both marginal and conditional distributions across two project. $\mathcal{M}_d$ implies the distribution difference of different class between different projects. $\mathcal{G}$ represents the term of manifold regularization. Ω is the term structural risk. In addition μ, η and λ are the trade-off parameters.

C. Transferability in LPDA

Drawing on previous literatures [15], [18], we adopt maximum mean discrepancy (MMD) [38] to measure both the marginal distributions and conditional distribution distances between source and target projects. The marginal distributions distance can be achieved as follows:

$$\begin{aligned} M_{\text{mar}}(\mathbf{X}_s, \mathbf{X}_t) &= \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \phi(\mathbf{x}_i^s) - \frac{1}{n_t} \sum_{j=1}^{n_t} \phi(\mathbf{x}_j^t) \right\|_{\mathcal{H}}^2 \\ &= \text{tr}(\mathbf{A}^\top \mathbf{K} \mathbf{M}_0 \mathbf{K}^\top \mathbf{A}), \end{aligned} \tag{2}$$

where, $\mathbf{X} = [\mathbf{X}_s, \mathbf{X}_t]$. $\mathbf{M}_0$ is the marginal distribution MMD matrix and it is computed as

$$(\mathbf{M}_0)_{ij} = \begin{cases} \frac{1}{n_s^2}, & \text{if} \quad \mathbf{x}_j, \mathbf{x}_i \in \mathbf{X}_s, \\ \frac{1}{n_s^2}, & \text{if} \quad \mathbf{x}_j, \mathbf{x}_i \in \mathbf{X}_t, \\ \frac{-1}{n_s n_t}, & \text{otherwise.} \end{cases} \tag{3}$$

Pseudo-labels in the target project are annotated by a classifier trained on the source project to represent conditional distribution. The conditional distributions distance can be

achieved as follows:

$$M_{\text{con}}\left(\mathbf{X}_s, \mathbf{X}_t\right) = \sum_{c=1}^{C} \left\| \frac{1}{n_{s,c}} \sum_{i=1}^{n_{s,c}} \phi(\mathbf{x}_i^{s,c}) - \frac{1}{n_{t,c}} \sum_{j=1}^{n_{t,c}} \phi(\mathbf{x}_j^{t,c}) \right\|_{\mathcal{H}}^2 = \operatorname{tr}(\mathbf{A}^\top \mathbf{K} \sum_{c=1}^{C} \mathbf{M}_c \mathbf{K}^\top \mathbf{A}), \tag{4}$$

where, $n_{s,c}$ and $n_{s,t}$ refer to the numbers of the c class instances of source and target projects, respectively. $\mathbf{M}_c$ is the conditional distribution MMD matrix with the label c, and it is computed as

$$(\mathbf{M}_c)_{ij} = \begin{cases} \frac{1}{n_{s,c}^2}, & \text{if} \quad \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_s^c, \\ \frac{1}{n_{t,c}^2}, & \text{if} \quad \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}_t^c, \\ \frac{-1}{n_{s,c} n_{t,c}}, & \text{if} \begin{cases} \mathbf{x}_i \in \mathbf{X}_s^c, \mathbf{x}_j \in \mathbf{X}_t^c, \\ \mathbf{x}_i \in \mathbf{X}_t^c, \mathbf{x}_j \in \mathbf{X}_s^c, \end{cases} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Here, M_{mar} plus M_{con} is rewritten as the follows:

$$\mathcal{M}_t\left(\mathbf{X}_s, \mathbf{X}_t, \mathbf{A}\right) = M_{\text{mar}} + M_{\text{con}} = \operatorname{tr}(\mathbf{A}^\top \mathbf{X} \sum_{c=0}^{C} \mathbf{M}_c \mathbf{X}^\top \mathbf{A}). \tag{6}$$

This section considers the transferability between different projects from both marginal and conditional distributions perspectives. However, only considering the transferability might not be sufficient for better prediction performance.

D. Discriminability in LPDA

In this section, we explore the discriminability between defective and no-defective classes by leveraging class-wise MMD to augment the distribution distance between different class, which can be achieved as follows:

$$\mathcal{M}_d\left(\mathbf{X}_s, \mathbf{X}_t, \mathbf{A}\right) = \sum_{c=1}^{C} \sum_{\widehat{c} \neq c} \left\| \frac{1}{n_{s,c}} \sum_{i=1}^{n_{s,c}} \phi(\mathbf{x}_i^{s,c}) - \frac{1}{n_{t,\widehat{c}}} \sum_{j=1}^{n_{t,\widehat{c}}} \phi(\mathbf{x}_j^{t,\widehat{c}}) \right\|_{\mathcal{H}}^2. \tag{7}$$

In order to facilitate the calculation, we introduce a one-hot coding label matrix to calculate the class-wise MMD based on the literature [45]. In particular, the source project and target project one-hot coding label matrices are written as $\mathbf{Y}_s = [c_{s,1}, ..., c_{s,n_s}]$ and $\widehat{\mathbf{Y}}_t = [c_{s,1}, ..., c_{s,n_s}]$. Two matrices are defined as follows

$$\mathbf{U}_s = \frac{1}{n_s} \left[\mathbf{Y}_s(:,1) \bullet (C-1), \ldots, \mathbf{Y}_s(:,C) \bullet (C-1) \right],$$
$$\mathbf{U}_t = \frac{1}{n_t} \left[\widehat{\mathbf{Y}}_t(:,1:C)_{\widehat{c} \neq 1}, \ldots, \widehat{\mathbf{Y}}_t(:,1:C)_{\widehat{c} \neq C} \right],$$

where, $\mathbf{Y}_s$ and $\widehat{\mathbf{Y}}_t$ denote the $c-th$ column of $\mathbf{Y}_s$ and $\widehat{\mathbf{Y}}_t$, respectively. $\mathbf{Y}_s(:,c) \bullet (C-1)$ denotes that $\mathbf{Y}_s(:,c)$ is repeated $C-1$ times. The Eq. (7) can be further expressed to matrix form as the follows:

$$\mathcal{M}_d\left(\mathbf{X}_s, \mathbf{X}_t, \mathbf{A}\right) = \operatorname{tr}(\mathbf{A}^\top \mathbf{X} \mathbf{U} \mathbf{X}^\top \mathbf{A}), \tag{8}$$

where,

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_s \mathbf{U}_s^\top & -\mathbf{U}_s \mathbf{U}_t^\top \\ -\mathbf{U}_t \mathbf{U}_s^\top & \mathbf{U}_t \mathbf{U}_t^\top \end{bmatrix}. \tag{9}$$

In this article, $\mathcal{M}_t$ and $\mathcal{M}_s$ are integrated and defined as discriminant distribution distance as

$$D(\mathbf{X}_s, \mathbf{X}_t) = \mathcal{M}_t\left(\mathbf{X}_s, \mathbf{X}_t, \mathbf{A}\right) - \mu \mathcal{M}_d\left(\mathbf{X}_s, \mathbf{K}_t, \mathbf{A}\right) = \operatorname{tr}\left(\mathbf{A}^\top \mathbf{X} \left(\sum_{c=0}^{C} \mathbf{M}_c - \mu \mathbf{U} \right) \mathbf{X}^\top \mathbf{A} \right). \tag{10}$$

Discriminant distribution alignment strategy is used to enhance perdition performance by Eq. (10) can to improve the transferability and discriminability.

E. Local Structure Preserving

From a geometric perspective, if two instances in intra-class are close in the intrinsic geometry of the data distribution, then their transforming should also be close [46]. However, the transformed features cannot well preserve original local relationship consistency. For example, the distance between two instances in intra-class but from the different projects may be expanded after feature transformation, resulting in this distance being longer than the distance between two instances in different classes but from the same projects. In an attempt to solve these problems, we ensure that instances in intra-class stay close after feature transformation and keep instances in inter-class far from each other. To preserve the local structure in the transforming subspace, we have defined two embedding graphs (reward graph penalty graph)as expressed as below.

Reward Graph: $\mathbf{x}_i^c$ is connected with $\mathbf{x}_j^c$, where $\mathbf{x}_j^c$ is one of the k nearest neighbors of $\mathbf{x}_i^c$, $\mathbf{x}_i^c \in \mathbf{X}^c$ and $\mathbf{x}_j^c \in \mathbf{X}^c$.

Penalty Graph: $\mathbf{x}_i^c$ is connected with $\mathbf{x}_j^{\widehat{c}}$, where $\mathbf{x}_j^{\widehat{c}}$ is one of the k nearest neighbors of $\mathbf{x}_i^c$, $\mathbf{x}_i^c \in \mathbf{X}^c$ and $\mathbf{x}_j^{\widehat{c}} \notin \mathbf{X}^c$ $(c \neq \widehat{c})$.

The connection edges between the nodes of the two graph are assigned weights

$$\mathbf{W}_{ij} = \begin{cases} \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2} \right), & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are conencted} \\ 0, & \text{otherwise.} \end{cases}$$

We denote the pre-defined reward and penalty weight matrices as $\mathbf{W}^r$ and $\mathbf{W}^p$, respectively. The intra-class and inter-class scatter matrices are respectively defined as follows ($\mathbf{S}_b$ denotes the inter-class scatter matrix, and $\mathbf{S}_w$ denotes the intra-class scatter matrix):

$$\mathbf{S}_w = \frac{1}{2} \sum_{i=1}^{n_c} \sum_{j=1}^{n_c} W_{ij}^r \left\| \mathbf{x}_i^c - \mathbf{x}_j^c \right\|^2 = \operatorname{tr}(\mathbf{A}^\top \mathbf{X} \mathbf{L}^r \mathbf{X}^\top \mathbf{A}), \tag{11}$$

$$\mathbf{S}_b = \frac{1}{2} \sum_{i=1}^{n_c} \sum_{j=1}^{n-n_c} W_{ij}^p \left\| \mathbf{x}_i^c - \mathbf{x}_j^{\widehat{c}} \right\|^2 = \operatorname{tr}(\mathbf{A}^\top \mathbf{X} \mathbf{L}^p \mathbf{X}^\top \mathbf{A}), \tag{12}$$

where, $\mathbf{L}^r$ and $\mathbf{L}^p$ respectively represent the reward Laplacian graph and the penalty Laplacian graph. ($\mathbf{L}^r = \mathbf{D}^r - \mathbf{W}^r$, $\mathbf{L}^p = \mathbf{D}^p - \mathbf{W}^p$). $\mathbf{D}_r$ and $\mathbf{D}_p$ are diagonal matrices whose each diagonal entry is $D_{ii}^r = \sum_j W_{ij}^r$ and $D_{ii}^p = \sum_j W_{ij}^p$, respectively. The geometric structure regularization can be expressed as follows:

$$\mathcal{G}\left(\mathbf{X}_s, \mathbf{X}_t, \mathbf{A}\right) = \mathbf{S}_w - \mathbf{S}_b = \operatorname{tr}(\mathbf{A}^\top \mathbf{X} (\mathbf{L}^r - \mathbf{L}^p) \mathbf{X}^\top \mathbf{A}). \tag{13}$$

F. Subspace Learning

Finally, the objective function in Eq. (1) can be reformulated byto maximize preserve the geometric structure of project instances.

$$\underset{\mathbf{A},\Phi}{\mathcal{L}} = \min \operatorname{tr}\left(\mathbf{A}^\top \mathbf{X}\left(\sum_{c=0}^{C}\mathbf{M}_c - \mu\mathbf{U} + \eta(\mathbf{L}^r - \mathbf{L}^p)\right)\mathbf{X}^\top\mathbf{A}\right) + \lambda\|\mathbf{A}\|_F^2, \quad \text{s.t.} \quad \mathbf{A}^\top\mathbf{X}\mathbf{H}\mathbf{X}^\top\mathbf{A} = \mathbf{I}, \tag{14}$$

where, $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}$ is the centering matrix to avoid trivial solutions, and $\mathbf{I} \in \mathbb{R}^{n\times n}$ is the identity matrix. According to the constrained optimization, deriving the Lagrange function finds solution of problem Eq. (14). The Lagrange function is:

$$\underset{\mathbf{A},\Phi}{\mathcal{L}} = \min \operatorname{tr}\left(\mathbf{A}^\top\left(\mathbf{X}\left(\sum_{c=0}^{C}\mathbf{M}_c - \mu\mathbf{U} + \eta(\mathbf{L}^r - \mathbf{L}^p)\right)\mathbf{X}^\top + \lambda\mathbf{I}\right)\mathbf{A}\right) + \operatorname{tr}\left(\left(\mathbf{I} - \mathbf{A}^\top\mathbf{X}\mathbf{H}\mathbf{X}^\top\mathbf{A}\right)\Phi\right), \tag{15}$$

where, $\Phi = diag(\phi_1, ..., \phi_d)$ is a diagonal matrix with the Lagrange multipliers. By setting the derivative of Eq. (15) $\frac{\partial\mathcal{L}}{\partial\mathbf{A}} = 0$, the optimization can be solved as a eigen-decomposition problem displayed as follows:

$$\left(\mathbf{X}\left(\sum_{c=0}^{C}\mathbf{M}_c - \mu\mathbf{U} + \eta(\mathbf{L}^r - \mathbf{L}^p)\right)\mathbf{X}^\top + \lambda\mathbf{I}\right)\mathbf{A} = \mathbf{X}\mathbf{H}\mathbf{X}^\top\mathbf{A}\Phi. \tag{16}$$

By taking the first d_0 smallest eigenvectors, the optimal solution of transformation matrix $\mathbf{A}$ is obtained. We can acquire the new representation $\mathbf{Z}_s = \mathbf{A}^\top\mathbf{X}_s$ and $\mathbf{Z}_t = \mathbf{A}^\top\mathbf{X}_t$. In summary, Algorithm 1 presents the pseudo code of the proposed LPDA approach.

Algorithm 1 Locality Preserving and Distribution Alignment (LPDA)

Input: Labeled source project: $\mathcal{D}_s = \{\mathbf{X}_s, \mathbf{Y}_s\}$; Unlabeled target project: $\mathcal{D}_t = \{\mathbf{X}_t\}$; Subspace dimension d_0; The number of iterations T; Number of nearest neighbors k; The trade-off hyper-parameters μ, η and λ.

Initialize pseudo labels $\widehat{\mathbf{Y}}_t$ of target project by using the classification model trained the source project.
Let $\mathbf{X} = [\mathbf{X}_s, \mathbf{X}_t]$;
for *z=1:T* **do**
  Construct MDD matrices $\mathbf{M}_0$ and $\mathbf{M}_c$ by Eq. (3) and Eq. (5);
  Construct discriminability matrix $\mathbf{U}$ by Eq. (9);
  Construct the reward Laplacian graph $\mathbf{L}^r$ and penalty Laplacian graph: $\mathbf{L}^p$;
  Solve Eq. (16) and take the d_0 smallest eigenvectors to construct $\mathbf{A}$;
  Obtain the source feature presentations $\mathbf{Z}_s = \mathbf{A}^\top\mathbf{X}_s$;
  Obtain the source feature presentations $\mathbf{Z}_t = \mathbf{A}^\top\mathbf{X}_t$;
  Train a classifier $f(\cdot)$ by using the source features presentations $\mathbf{Z}_s$ and labels $\mathbf{Y}_s$;
  Update the target pseudo labels $\widehat{\mathbf{Y}}_t$ by using the classifier $f(\cdot)$.

Output: The transformation matrix $\mathbf{A}$.

G. Computational Complexity

In this subsection, we present an analysis to time complexity of the proposed approach in Algorithm 1. The computational cost of solving eigen-decomposition problem is $O(T\times d_0\times d^2)$, of constructing the MMD matrices, Laplacian graphs and the two discriminability matrix is $O(T\times C\times n^2)$, and of all other steps is $O(T\times d\times n)$. Thus, the overall computational complexity is $O(T\times d_0\times d^2 + 3\times T\times C\times n^2 + T\times d\times n)$.

TABLE I. ESSENTIAL INFORMATION OF THE SOFTWARE PROJECTS APPLIED IN THIS ARTICLE

Dataset	Project	# of metrics	# of total instances	% rate defective
Promise	ant-1.7	20	745	22.28
	ivyv-2.0	20	352	11.36
	jedit-4.1	20	312	25.32
	log4j-1.0	20	135	25.19
	lucene-2.2	20	247	58.30
	pio-2.0	20	314	11.78
	synapse-1.1	20	222	27.03
	tomcat	20	858	8.97
	xerces-1.4	20	588	74.32
NASA	CM1	37	327	12.84
	MW1	37	253	10.67
	PC1	37	705	8.65
	PC3	37	1077	12.44
	PC4	37	1287	13.75
AEEEM	EQ	61	324	39.81
	JDT	61	997	20.66
	LC	61	691	9.26
	ML	61	1862	13.16
	PDE	61	1497	13.96
ReLink	Apache	26	194	50.52
	Safe	26	56	39.29
	ZXing	26	399	29.57

IV. EXPERIMENTAL SETUP

A. Benchmark Datasets

To assess the put forward method, a total of 22 publicly software projects from four different software repositories, including AEEEM [47], NASA [48], Relink [49] and PROMISE are applied in our experiments. Table I offers comprehensive information about these projects.

AEEEM includes five software project [47]. Software module (instance) granularity is classified in AEEEM. Each instance includes 61 metrics (features), among them five entropy-of-change metrics, 17 code metrics, five previous-defect metrics, 17 entropy-of-code metrics and 17 churn-of-code metrics.

NASA is the most popular software defect data in previous studies. Each project signifies a software system, encompassing static code metrics along with associated defect labels. The static code metrics encompass attributes like McCabe, Halstead, lines of code, among others, and are valuable for predicting software quality and defects. The static code metrics encompass McCabe complexity, Halstead intricacy, code line count, and similar factors. These measurements offer valuable insights into software quality and predicting defects. In the research, we selected five projects that shared common feature spaces and merged them to create 20 cross-project predictive tasks. *Promise* is collected by Jureczko and Madeyski [50] and , comprises 20 class-level metrics, including CK metrics and QMOOD metrics. In the article, we selected 10 open projects and then used these projects to combine 90 cross-project prediction tasks.

ReLink is denoted by Wu et al. containing three open

projects (i.e Apache, Safe and ZXing) They can be combined into cross-project prediction six tasks.

B. Performance Indicators

In this article, applying the four performance indicators, including F-measure, Balance, MCC and AUC evaluate our method and comparison methods. A software defect task typically yields one of four typical output results:

True Positive (TP): Correctly predicted defective instances.

True Negative (TN): Correctly predicted non-defective instances.

False Positive (FP): Incorrectly predicted defective instances.

False Negative (FN): Incorrectly predicted non-defective instances.

PD (aka. *recall or sensitivity*) determines the defective instances correctly predicted. The higher PD, the lower the cost generated by type II mis-classifications. *PF*(aka. *false positive rate*) is the ratio of non-defective instances that are incorrectly predicted.*Precision* is the ratio of correctly predicted instances that are defective They are denoted as the follows.

$$PD = Recall = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}};$$
$$PF = \frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}};$$
$$Precision = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}.$$

F-measure is a harmonic mean of *Precision* and *Recall*, which is denoted as

$$F\text{-}measure = \frac{(1+\theta^2) \times Recall \times Precision}{Recall + \theta^2 \times Precision}.$$

where, θerves as a bias parameter that determines the relative significance of *Recall* and *Precision*. There are three F-measure variants, i.e., $\mathbf{F_1}$ ($\theta = 1$) treats precision and recall equally, $\mathbf{F_{0.5}}$ ($\theta = 0.5$) prefers precision, and $\mathbf{F_2}$ ($\theta = 2$) prefers recall. There are two types of error in defect prediction. The first type of misclassification is the prediction of non-defect instances as defect instances. The second type of misclassification is when a defect instance is predicted to be a non-defect instance. The cost of the latter error is higher than that of the former error. Defect prediction mainly concerns finding as many defective instances as possible, which is consistent with the evaluation of performance measurement bias to recall. Thus, defect prediction more emphasize that Recall is more important than Precision(θ is set as 2).

Balance is the normalized Euclidean distance from the ideal point (1,0) to the actual point (*PD, PF*) in the ROC curve[51], [52], which is denoted as

$$Balance = 1 - \frac{\sqrt{(1-PD)^2 + (0-PF)^2}}{\sqrt{2}}.$$

MCC (Matthews Correlation Coefficient) is to measure the correlation coefficient between the actual and predicted outputs, which is denoted as:

$$MCC = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}.$$

AUC(Area Under the Curve) The Area Under the Curve (AUC) refers to the area under the receiver operating characteristic curve (ROC). AUC is a comprehensive metric that effectively captures the trade-off and provides a better reflection of the overall performance of a prediction model.

C. Research Questions

To assess the predictive performance of the proposed LPDA, we delve into three research questions, in depth.

RQ1: Does LPDA perform better than the instances-based CPDP approaches?

As baselines for addressing this question, we employed five instance transfer approaches, which include ALL, NN-Filter, TNB, Peter-Filter, and DTB. Among these, TNB and DTB involve the reweighting of instances to mitigate the adverse influence of irrelevant cross-project data. In contrast, NN-Filter and Peter-Filter focus on filtering instances from the source project that are similar to the target project. These methods do not change the original feature space. Unlike these methods, the proposed method transfers the feature spaces while exploiting all the instances in the training step to avoid information loss. This research question is designed to investigate whether LPDA is superior to the instances-based CPDP methods in terms of CPDP performance improvement.

RQ2: Is LPDA more effective than the domain adaptation based CPDP methods?

We attempted to improve the CPDP performance from two aspects: the transferability between projects and the discriminability between class. Typically, domain adaptation-based CPDP methods investigate transferability by assuming that the source and target projects share a common distinguishing boundary. However, although the distribution gap between the two projects is narrowed after feature transformation, the instances from different classes are too close to be classified accurately near the decision boundary. The proposed LPDA method simultaneously explores the transferability and discriminability for CPDP tasks. This research question is designed to investigate whether the method considering both the transferability and discriminability is better able to improve the CPDP performance compared with other transfer learning methods.

RQ3: How do LPDA components have affect the prediction performance?

Since locality-preserving projection and distribution alignment are used in the proposed method, this research question is designed to investigate whether the components (i.e. transferable distribution alignment, discriminant distribution alignment and locality-preserving projection) can affect the prediction performance.

D. Statistical Testing

To better illustrate the effectiveness of the proposed method, we employ two statistical testing methods, namely the Wilcoxon signed-rank test and the Scott-Knott ESD test.

The Wilcoxon signed-rank test used to determine whether a significant difference exists between our method and each baseline for each cross-project task. Data distributions are identical without the assumption that they follow the normal distribution. Additionally, we employ the Win/Tie/Lose (W/T/L for brevity) evaluation to assess how many cross-project tasks our method can enhance in comparison to each baseline. Each entry's W/T/L implies that our method outperforms W cross-project tasks, ties on T cross-project tasks, and loses on L cross-project tasks.

The Scott-Knott ESD test is an expansion of the statistical methodology developed by Scott-Knott, which employs hierarchical cluster analysis to categorize a set of evaluation metrics into distinct, non-overlapping groups with statistically significant differences. This test consists of two stages: (1) the identification of a partition that maximizes the mean between different groups, and (2) either the separation into two distinct groups or the merging of any two groups with statistically significant differences and a negligible effect size into a single group. For a comprehensive explanation of the Scott-Knott ESD test, please refer to [53].

E. Experimental Settings

A total of 23 projects from four different repositories including AEEEM (5 projects), NASA (5 projects), ReLink (3 projects) and Promise (9 projects) are used in the paper. We we first identify all cross-project tasks in NASA, AEEEM, Relink and Promise. One project is selected as the target project, and the other projects from the same repository as the source. For example, when EQ is selected as the target project, the other projects separately are use as the source project. There are four cross-project tasks: JDT⇒EQ, LC ⇒EQ, ML⇒EQ, and PDE⇒ EQ. In total, there are 118 (9× 8+5×4 +5×4+3×2) cross-project tasks. We repeat each cross-project task 30 times and report the average values, each time we randomly select 90% of instances the source projects as the training set to train the model and all instances from the target project as test set, since a random selection of 90% of instances ensure that the training data are not consistently identical. In the article, Logistic Regression (LR) is selected as the foundational classifier. Due to its simplicity and effective performance in contrast to more intricate modeling methods, LR (Logistic Regression) has been a common choice in previous SDP research[18].

V. Experimental Results

A. Results for RQ1

To investigate the question, we apply some CPDP based-instance transferring methods as the baselines, including ALL, NN-Fifter, Petter-Fifter, TNB and DTB. ALL means that all the instances from the source project are used to train the prediction model without any instance filter and reweight process.

TABLE II. Average Values of Four Indicators for LPDA and Five Instance based CPDP Methods on the Different Dataset

Dataset	indicators	ALL	NN-Filter	Peter-Filter	TNB	DTB	LPDA
AEEEM	F-measure	0.404	0.423	0.409	0.532	0.486	**0.542**
	Balance	0.572	0.600	0.578	0.601	0.597	**0.696**
	MCC	0.204	0.233	0.239	0.248	0.276	**0.378**
	AUC	0.656	0.662	0.611	0.716	0.669	**0.748**
NASA	F-measure	0.358	0.417	0.317	0.418	0.393	**0.474**
	Balance	0.603	0.633	0.536	0.606	0.603	**0.670**
	MCC	0.179	0.186	0.193	0.202	0.206	**0.233**
	AUC	0.628	0.686	0.563	0.709	0.662	**0.724**
Promise	F-measure	0.442	0.482	0.498	0.459	0.457	**0.563**
	Balance	0.596	0.624	0.639	0.658	0.662	**0.702**
	MCC	0.250	0.316	0.256	0.322	0.259	**0.338**
	AUC	0.662	0.683	0.690	0.722	0.731	**0.746**
RELINK	F-measure	0.543	0.534	0.569	0.587	0.546	**0.767**
	Balance	0.607	0.640	0.638	0.595	0.637	**0.708**
	MCC	0.262	0.292	0.269	0.289	0.280	**0.320**
	AUC	0.669	0.700	0.702	0.641	0.701	**0.745**

TABLE III. Wilcoxon Signed-rank Test Results of LPDA Against Each Instance based CPDP Method

Dataset	Against(W/T/L)					
	indicators	ALL	NN-Filter	Peter-Filter	TNB	DTB
AEEEM	F-measure	20/0/0	19/1/0	19/1/0	14/2/4	15/3/2
	Balance	20/0/0	18/2/0	18/2/0	16/0/4	13/4/3
	MCC	19/0/1	19/0/1	18/0/2	17/1/3	14/2/4
	AUC	18/1/1	17/3/0	20/0/0	13/5/2	16/2/2
NSNA	F-measure	18/1/1	14/3/3	18/1/1	15/1/4	14/3/3
	Balance	15/1/4	13/2/5	17/2/1	16/1/3	17/0/3
	MCC	16/1/3	17/0/3	16/1/3	14/1/5	14/1/5
	AUC	16/2/2	12/2/5	19/0/1	8/5/7	18/2/0
Promise	F-measure	61/3/8	55/6/11	46/4/22	47/4/22	56/4/12
	Balance	64/4/4	57/4/11	54/4/14	46/12/14	53/6/13
	MCC	52/5/15	51/15/6	49/6/17	43/20/9	50/4/18
	AUC	56/3/13	53/4/15	48/8/16	44/6/22	40/6/26
RELINK	F-measure	6/0/0	6/0/0	6/0/0	6/0/0	6/0/0
	Balance	6/0/0	6/0/0	6/0/0	5/1/0	5/1/0
	MCC	6/0/0	5/1/0	6/0/0	5/0/1	5/1/0
	AUC	6/0/0	4/2/0	6/0/0	6/0/0	4/2/0
Total	F-measure	105/7/ 6	94/13/11	89/9/20	81/16/21	91/13/14
	Balance	105/8/5	94/11/13	95/11/12	83/17/9	88/14/16
	MCC	93/9/16	92/19/7	89/10/19	79/25/14	83/11/24
	AUC	96/9/13	86/15/17	93/11/14	71/19/28	78/15/25

According to the data presented in Table II, the proposed LPDA consistently outperforms the five baseline methods across various indicators on all datasets. For instance, on the AEEEM dataset, LPDA achieves an average F-measure value of 0.542, which represents a substantial improvement, ranging from 1.87% (in comparison to TNB) to a remarkable 34.06% (in comparison to ALL), with an average enhancement of 21.6%. In terms of the average Balance score (0.696) achieved by LPDA, improvements range from 15.77% (in comparison to TNB) to 21.68% (in comparison to ALL), with an average boost of 18.01%. Moreover, the average MCC value (0.376) obtained with LPDA demonstrates significant improvements, varying from 36.81% (in comparison to DTB) to an impressive 85.59% (in comparison to ALL), with an average enhancement of 59.04%. The average AUC value (0.748) also shows positive trends, with improvements ranging from 4.47% (in comparison to TNB) to 22.38% (in comparison to NN-Filter), averaging a substantial 12.85% improvement when contrasted with the five instance-based CPDP methods. Concerning the 20 cross-project pairs on the AEEEM dataset, data from Table III reveal that LPDA exhibits a statistically significant superiority in at least 13 of these cross-project pairs across all indicators. Conversely, it may perform less favorably on most of the four cross-project pairs. Fig. 1 demonstrates that the median values of all four indicators by LPDA higher than the five baseline methods. In particular, the median F-measure, Balance and AUC of the Relink dataset by LPDA are similar or even better than to the maximum value achieved by the five baseline methods.

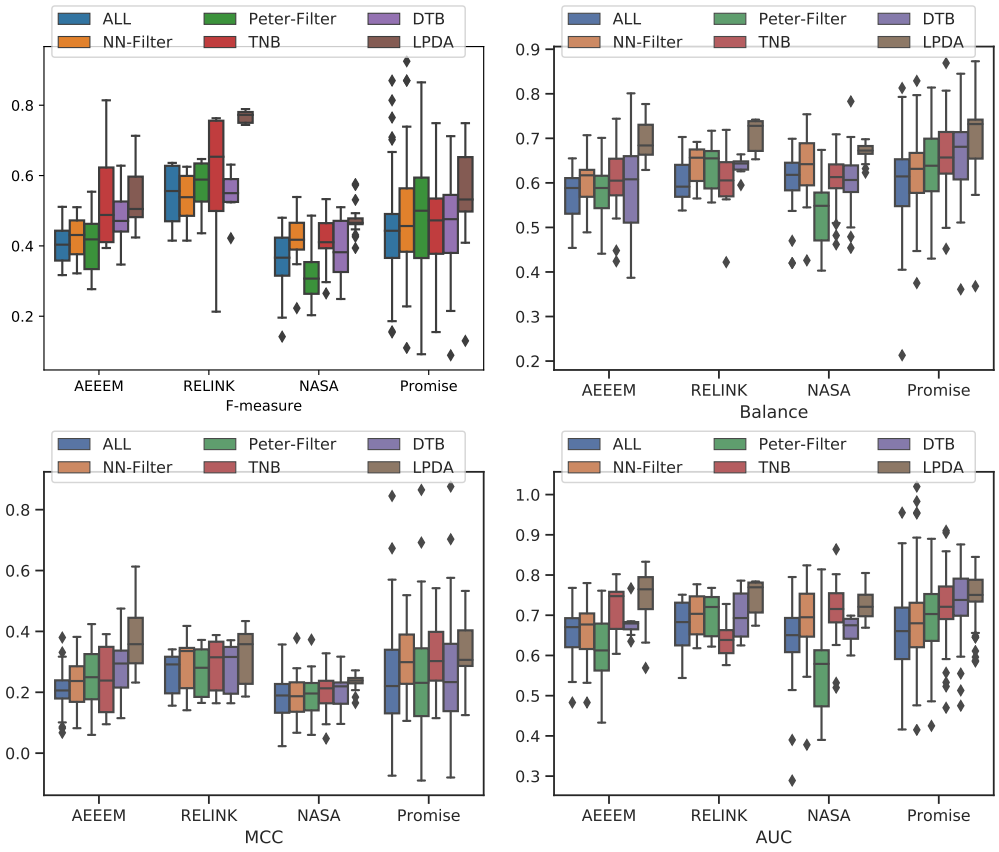


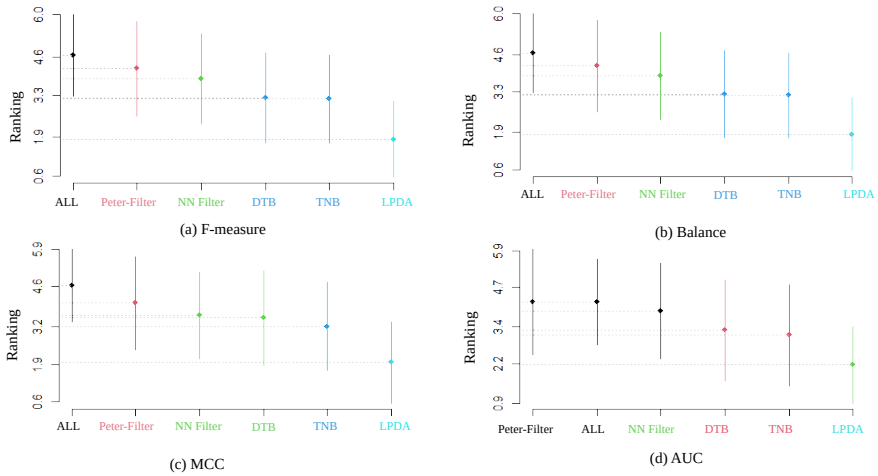
Fig. 1. Boxplots of f-measure, balance MCC, and AUC across all datasets for LPDA and the five instance-based CPDP methods.

On NASA dataset, the average F-measure value (0.474) by LPDA yields improvements between 13.40% (for TNB) and 49.45% (for Peter-Filter) with an average improvement of 25.92%, the average Balance value (0.670) by LPDA gains improvements between 5.77% (for NN Filter) and 24.89% (for Peter-Filter) with an average improvement of 12.65%, the average MCC value (0.233)by LPDA achieves improvements between 13.37% (for DTB) and 30.41%(for ALL) with an average improvement of 21.07%, and the average AUC value (0.724) gets improvements between 2.14% (for DTB) and 28.61% (for Pete-Filter) with an average improvement of 12.23% compared against the five instances-based CPDP methods. Three are also 20 cross-project pairs on NASA dataset. From Table III, the result shows that LPDA is significantly more accurate at least on 14 cross-project pairs and significantly less accurate at most on five cross-project pairs in terms of F-measure, Balance and MCC indicators. LPDA almost equal TNB (eight wins and seven losses) in terms of AUC. Compared to other the four baseline methods, LPDA is almost complete victory (at least 12 wins and at most 5 losses).

On the RELINE dataset, LPDA achieves an average F-measure value of 0.767, resulting in improvements ranging from 30.76% (compared to TNB) to 43.36% (compared to Peter-Filter), with an average improvement of 38.25%. The average Balance value, at 0.708 by LPDA, shows enhancements ranging from 10.64% (compared to NN-filter) to 18.98% (compared to TNB), with an average improvement of 13.66%. Furthermore, the average MCC value of 0.320 by LPDA demonstrates improvements ranging from 9.75% (compared to NN-filter) to 22.50% (compared to ALL), with an average improvement of 15.33%. The average AUC value of 0.745 also shows enhancements ranging from 6.16% (compared to Peter-Filter) to 16.16% (compared to DTB), with an average improvement of 9.27% when compared to the five instances-based CPDP methods. It is evident from Table III that LPDA outperforms the five baseline methods in terms of these four indicators, albeit it falls slightly short of them in certain aspects.

On Promise dataset, the average F-measure value (0.563) by LPDA yields improvements between 13.06% (for Peter-Filter) and 27.25% (for ALL) with an average improvement of 20.56%, the average Balance value (0.702) by LPDA gains improvements between 6.05% (for DTB) and 17.73% (for ALL) with an average improvement of 10.46%, the average MCC value (0.338) by LPDA achieves improvements between 4.84% (for TNB) and 34.90% (for ALL) with an average improvement of 21.84%, and the average AUC value (0.746) gets improvements between 2.08% (for DTB) and 12.72% (for ALL) with an average improvement of 7.07% compared against the five instances-based CPDP methods.

Table III shows the results of Wilcoxon signed-rank statistical test for each baseline method. By using the "W/T/L" evaluation, we can investigate 118 cross-project pairs in which LPDA can outperform other comparing method. As shown in the table, it is obvious that LPDA can achieve more positive



(a) F-measure (b) Balance (c) MCC (d) AUC

Fig. 2. The results of scott-knott ESD test in f-measure, balance MCC, and AUC across all datasets for LPDA and other instance-based methods. The smaller ranking, the better performance.

results in terms of F-measure,Balance, MCC and AUC measure indicators when compared with other competing methods. For example, LPDA has significant superiorities to TNB on 81/118 (81 out of 118 cross-project pairs) in F-measure, and 83, 79 and 71 in Balance, MCC, and AUC, respectively. Fig. 2(a)-(d) present the results of Scott-Knott ESD test for the proposed LPDA and five baseline methods in terms of F-measure, Balance,MCC, and AUC, respectively. The x-axis and y-axis represent the method and ranking, respectively. The smaller the ranking, the better the performance. Each method corresponds to a bar denoting the range of ranking of this method on all cross-project pair tasks. The dot in the bar indicates the average ranking value. Different colors denote different groups with statistically significant differences. From these figures, we can see that LDPA obtains the smallest average ranking and thus is categorized into the top group which do not include any baseline method in terms of the four indicators.

B. Results for RQ2

In order to address this question, we have chosen five domain adaptation-based methods. A concise overview of these baseline methods is provided below:

1) *TCA*: This method aims to match marginal distribution between two projects [38] .
2) *TCA+*: An TCA variation improved in previous research [17] add customized normalization rules before distribution matching.
3) *JDM*: This method matches the marginal and conditional distributions simultaneously [15].
4) *BDA*: BDA [18] takes into account both the marginal and conditional distributions and dynamically assigns varying weights to them.
5) *TPTL*: TPTL [39] selects the benefits of source project selection and use transfer learning to construct a SDP model.

Table IV presents the mean values of the four metrics for both LPDA and the five domain adaptation-based methods across the four datasets. Fig. 3 illustrates box-plots representing the four metrics for all six methods across the entire set of datasets.

TABLE IV. AVERAGE VALUES OF FOUR INDICATORS FOR LPDA AND FIVE DOMAIN ADAPTATION BASED CPDP METHODS ON THE DIFFERENT DATASET

Dataset	indicator	TCA	TCA+	JDM	BDA	TPTL	PLDA
AEEEM	F-measure	0.468	0.487	0.505	0.533	0.510	**0.542**
	Balance	0.671	0.645	0.674	0.689	0.647	**0.696**
	MCC	0.217	0.244	0.253	0.258	0.290	**0.378**
	AUC	0.701	0.716	0.725	0.703	0.729	**0.748**
NASA	F-measure	0.391	0.382	0.456	**0.484**	0.456	0.474
	Balance	0.651	0.572	0.646	0.666	0.660	**0.670**
	MCC	0.190	0.199	0.205	0.216	0.220	**0.233**
	AUC	0.657	0.675	0.705	0.707	**0.726**	0.724
Promise	F-measure	0.472	0.448	0.538	0.487	0.511	**0.563**
	Balance	0.656	0.678	0.693	0.699	**0.706**	0.702
	MCC	0.247	0.316	0.249	0.314	0.287	**0.338**
	AUC	0.668	0.677	0.693	0.690	0.705	**0.746**
RELINK	F-measure	0.636	0.639	0.632	0.643	0.703	**0.767**
	Balance	0.630	0.487	0.599	0.639	0.640	**0.708**
	MCC	0.271	0.306	0.282	0.306	0.295	**0.320**
	AUC	0.611	0.621	0.645	0.665	0.733	**0.745**

According to the data in Table IV, LPDA outperforms the five domain adaptation-based methods in all metrics when it comes to the AEEEM dataset, with higher average values.More specifically, LPDA compared with the five domain adaptation based methods achieves improvements of 1.55%–15.75% in F-measure , 0.92%–7.8% in Balance, 30.40%–74.42% in MCC, 2.26%–6.68% in AUC.

On the NASA dataset, the proposed LPDA achieves the best average values in terms of Balance and MCC, while BDA and TPTL achieve the best average values in terms of F-measure and AUC, respectively. More specifically, compared with the five baseline methods, LPDA achieves improvements ranging from 0.58% to 17.11% on Balance, and 5.92% to 22.99% on MCC, respectively. However, the average F-measure and AUC of LPDA are 1.95% and 0.2% lower than TNB and DTB, respectively. The results presented in Table IV show that average values in four evaluation indicators by

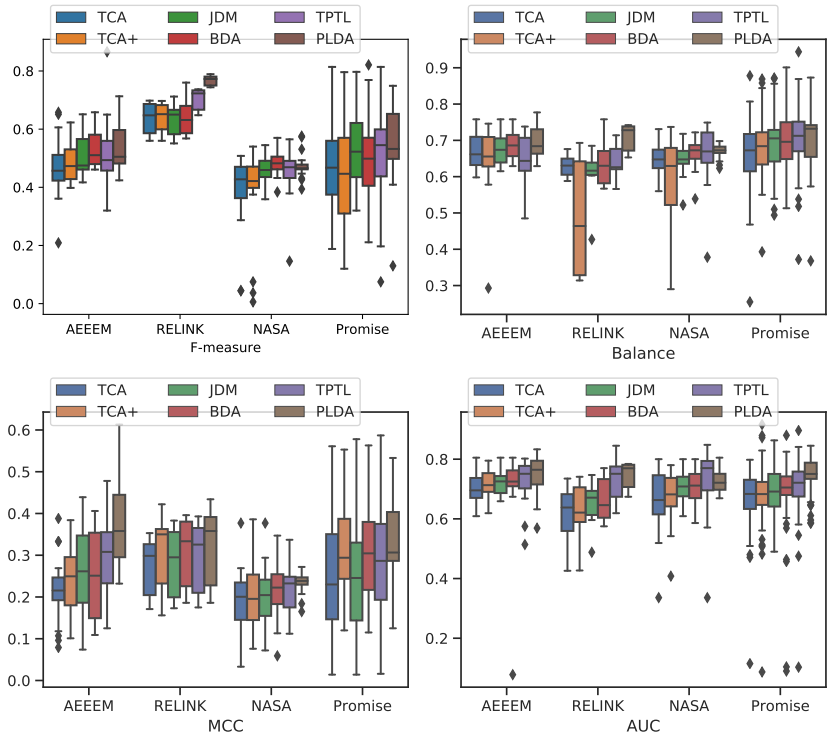


Fig. 3. Boxplots of f-measure, balance MCC, and AUC across all datasets for LPDA and the five domain adaptation based CPDP methods.

TABLE V. WILCOXON SIGNED-RANK TEST RESULTS OF LPDA AGAINST EACH DOMAIN ADAPTATION BASED CPDP METHOD

Dataset	Against(W/T/L)					
	indicators	TCA	TCA+	JDM	BDA	TPTL
AEEEM	F-measure	13/4/3	12/3/5	10/2/8	11/2/7	8/3/9
	Balance	14/2/4	14/3/3	13/3/4	8/9/3	14/3/3
	MCC	19/0/1	19/0/1	17/1/2	17/1/2	13/2/5
	AUC	18/1/1	14/4/2	13/4/3	11/4/5	13/4/3
NSNA	F-measure	13/4/3	12/3/5	10/2/8	11/2/7	8/3/9
	Balance	11/5/4	13/2/5	12/4/4	8/6/6	8/3/9
	MCC	14/3/3	14/3/3	13/4/3	11/4/5	9/5/6
	AUC	12/5/3	14/1/5	11/5/4	10/6/4	13/1/6
Promise	F-measure	54/1/17	50/1/21	39/2/31	49/7/16	44/3/25
	Balance	48/10/14	47/4/21	40/6/26	32/9/31	35/7/30
	MCC	55/2/15	45/15/12	52/4/16	38/19/15	44/5/23
	AUC	59/2/11	51/5/16	52/4/16	49/2/21	45/7/20
RELINK	F-measure	5/1/0	4/0/2	5/1/0	3/2/1	3/2/1
	Balance	6/0/0	6/0/0	6/0/0	5/1/0	5/1/0
	MCC	6/0/0	5/1/0	6/0/0	5/0/1	5/1/0
	AUC	6/0/0	6/0/0	6/0/0	6/0/0	3/2/1
Total	F-measure	90/10/18	84/8/26	71/8/39	71/18/29	72/13/33
	Balance	79/20/19	80/12/26	71/16/31	53/27/38	63/16/39
	MCC	93/9/16	82/21/15	87/13/18	69/29/20	69/17/32
	AUC	95/11/12	85/13/20	82/16/20	76/15/27	67/17/34

the proposed LPDA are the best average values on the Relink dataset. More specifically, compared with the five baseline methods, average value by LPDA gains the improvement of 9.17%-21.34% in terms of F-measure, of 10.53%–45.47% in terms Balance, of 4.61%–18.04% in terms MCC, and of 1.61%–21.82% in terms AUC. On the Promise dataset, the proposed LPDA achieves the best average values in terms of F-measure, MCC and AUC, while TPTL achieves the best average values in terms of Balance. To be specific, compared with the five baseline methods, LPDA achieves improvements ranging from 4.69% to 25.56% on F-measure, 6.97% to 36.79% on MCC, and 5.83% to 11.69% on AUC, respectively. However, the average value by LPDA is 0.58% lower than the best average value (for TPTL) among the five baseline methods in terms of Balance. From Table V, It is obvious that LPDA has more than 53 wins over Wilcoxon signed-rank test in any evaluation indicator. Taking TCA+ as an example, the "W/T/L" results show that LDPA has statistically significant improvements of 84/118 (84 out of 118 cross-project pair prediction), 80/118, 82/118 and 85/118 in F-measure, Balance MCC and AUC, respectively.

Fig. 3 depicts the boxplots of four indicators for the six methods on four datasets. This figure illustrates that the median values of all four indicators achieved by LPDA surpass those obtained by the five baseline methods on both the AEEEM and Relink datasets. On the NASA dataset, the median values of all four indicators by LPDA are not superior to these by the TPTL. On the Promise dataset, the values of Balance indicators

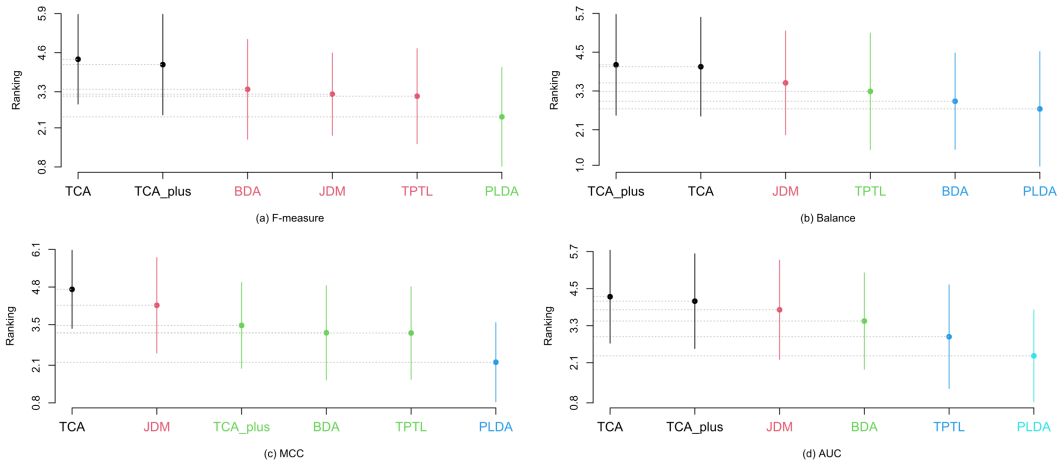


Fig. 4. The results of scott-knott ESD test in f-measure, balance MCC, and AUC across all datasets for LPDA and the five domain adaptation based CPDP methods.

by LPDA are slightly weaker than that by the TPTL.

Moreover, in Fig. 4(a)-(c), the outcomes of the Scott–Knott ESD test are presented for LPDA and the five domain adaptation-based methods across 118 cross-project pairs, considering F-measure, Balance, MCC, and AUC. The figures reveal that LPDA consistently achieves the lowest average ranking and is clearly separated into a distinct group concerning the four evaluation metrics. This indicates that LPDA significantly outperforms domain adaptation-based CPDP methods.

C. Results for RQ3

TABLE VI. AVERAGE VALUES OF FOUR INDICATORS FOR LPDA AND IT'S VARIANTS ON THE DIFFERENT DATASET

Dataset	indicator	LPDA_TA	LPDA_DA	LPDA_noDA	LPDA_noLP	PLDA
AEEEM	F-measure	0.474	0.498	0.496	0.504	**0.542**
	Balance	0.639	0.667	0.668	0.677	**0.696**
	MCC	0.336	0.363	0.358	0.361	**0.378**
	AUC	0.687	0.727	0.725	0.728	**0.748**
NASA	F-measure	0.389	0.399	0.392	0.413	**0.474**
	Balance	0.608	0.647	0.639	0.654	**0.670**
	MCC	0.214	0.212	0.212	0.228	**0.233**
	AUC	0.707	0.707	0.706	0.708	**0.724**
RELINK	F-measure	0.527	0.546	0.538	0.564	**0.767**
	Balance	0.687	0.670	0.701	0.666	**0.708**
	MCC	0.291	0.310	0.313	0.306	**0.320**
	AUC	0.713	0.717	0.717	0.732	**0.745**
Promise	F-measure	0.419	0.432	0.437	0.456	**0.563**
	Balance	0.662	0.666	0.683	0.684	**0.702**
	MCC	0.320	0.324	0.323	0.330	**0.338**
	AUC	0.719	0.728	0.727	0.729	**0.746**

LPDA has three key components, including locality preserving, transferable distribution transferable distribution alignment and discriminant distribution alignment. To investigate whether the proposed LPDA approach is more effective than other combinations of these three components, we specially conduct experiments to study the design of these components. We separately conduct LPDA_TA with only transferable distribution alignment (without locality preserving and discriminant distribution alignment), which is referred to as LPDA_TA, LPDA without discriminant distribution alignment, which is referred to as LPDA_noDA and LPDA without locality preserving, which is referred to as LPDA_noLP and LPDA with only discriminant distribution alignment which is referred to as LPDA_DA.

TABLE VII. WILCOXON SIGNED-RANK TEST RESULTS OF LPDA AGAINST EACH VARIANT METHOD

Dataset	Against(W/T/L)				
	indicators	LPDA_TA	LPDA_DA	LPDA_noDA	LPDA_noLP
AEEEM	F-measure	17/1/2	14/2/4	16/1/3	16/1/3
	Balance	13/2/5	14/5/1	15/4/1	12/4/4
	MCC	13/3/4	10/8/2	14/6/0	12/5/3
	AUC	14/1/5	15/0/5	11/6/3	9/4/7
NSNA	F-measure	17/1/2	16/3/1	17/1/2	14/1/5
	Balance	16/2/2	11/3/6	16/2/2	10/6/4
	MCC	13/5/2	14/5/1	17/2/1	10/7/3
	AUC	11/4/5	12/3/5	12/4/4	12/3/5
Promise	F-measure	56/5/11	58/3/11	55/3/11	50/8/14
	Balance	48/10/14	47/4/21	40/6/26	32/9/31
	MCC	47/8/17	42/23/7	41/28/3	35/32/5
	AUC	45/16/11	40/13/19	43/15/14	43/16/13
RELINK	F-measure	6/0/0	6/0/0	6/0/0	5/1/0
	Balance	4/1/1	6/0/0	4/0/2	6/0/0
	MCC	4/1/1	3/2/1	2/3/1	5/1/0
	AUC	5/0/1	4/1/1	5/0/1	4/1/1
Total	F-measure	96/10/15	94/11/16	94/9/18	85/14/22
	Balance	82/22/17	76/22/23	81/28/12	70/32/19
	MCC	77/29/15	69/41/11	74/42/5	62/48/11
	AUC	75/24/22	71/20/30	71/28/22	68/27/26

Table VI shows the average values of F-measure, Balance, MCC and AUC on four different datasets of LPDA_TA, LPDA_DA, LPDA_noDA, PLDA_noLS and LPDA on Relink dataset, AEEEM dataset, NASA dataset and Promise dataset. In these table, the best values in each row are in bold. Fig. 5 depicts the boxplots of four indicators for the five methods. Table VII shows the results of Wilcoxon signed-rank statistical test. We present the following findings in Table VI, Table VII and Fig. 5. The results on the AEEEM dataset show that the proposed LDPA performs better than its four variants. To be specific, the improvement of LPDA over LPDA_TA, LPDA_DA, LPDA_noDA and PLDA_noLS on average is at least by 8.83%, 4.13%, 4.21%, 2.72% and at most by 14.38%, 8.87%, 12.48%, 8.86% in terms of F-measure, Balance, MCC and AUC, respectively. The results of Wilcoxon signed-rank statistical test shows that LPDA achieves more than 12 improvements with reference to Balance and F-measure , more than 10 improvements with reference to MCC, and more than nine improvements with reference to AUC, respectively.

On NASA dataset, the improvement of LPDA over LPDA_TA, LPDA_DA, LPDA_noDA and PLDA_noLS on average is at least by 14.69%, 2.38%, 2.26% , 2.27% and at most by 21.95%, 10.19%, 10.18%, 2.56% with regard to F-measure, Balance, MCC, and AUC, respectively. The results

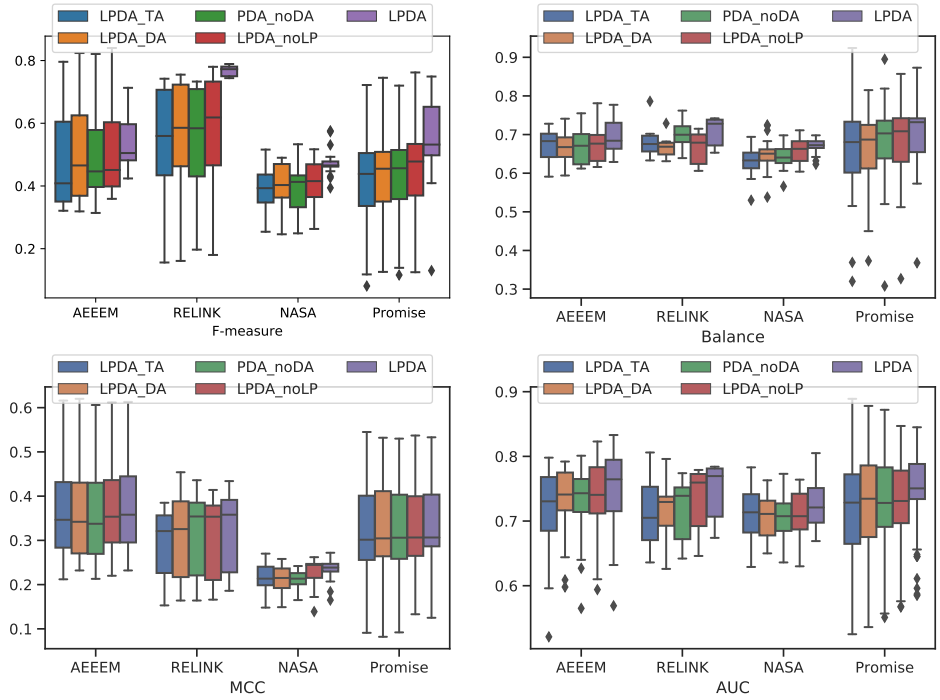


Fig. 5. Boxplots of f-measure, balance MCC, and AUC across all datasets for LPDA and it's variant methods.

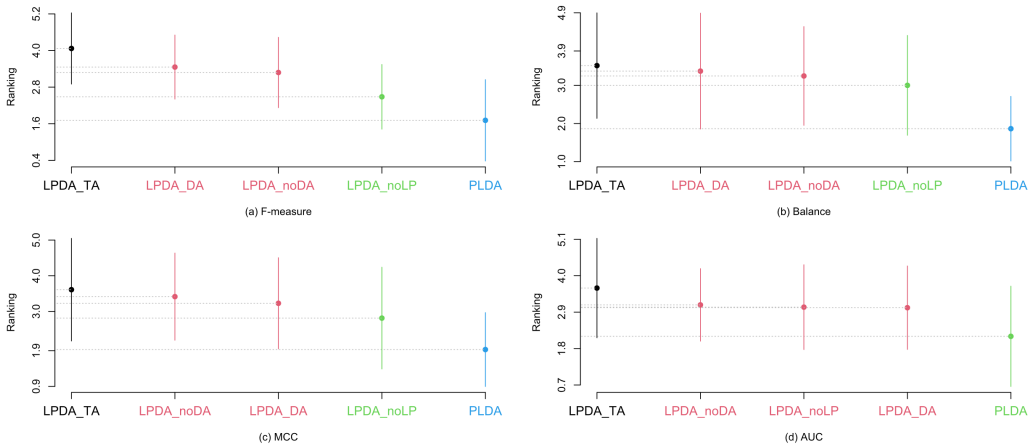


Fig. 6. The results of scott-knott ESD test in f-measure, balance MCC, and AUC across all datasets for LPDA and it's variant methods. The smaller ranking, the better performance.

of Wilcoxon signed-rank statistical test shows that LPDA achieves more than 10 improvements with reference to Balance and MCC , more than 14 improvements with reference to F-measure, and more than 11 improvements with reference to AUC respectively.

On Relink dataset, the improvement of our method over LPDA_TA, LPDA_DA, LPDA_noDA and PLDA_noLS on average is at least by 36.11%, 1.04%, 2.33% , 1.76% and at most by 45.51%, 6.38%, 10.28%, 4.9% with regard to F-measure, Balance, MCC, and AUC, respectively. The results of Wilcoxon signed-rank statistical test shows that LPDA achieves more than four improvements with reference to Balance, AUC and F-measure. However, MCC of LPDA only wins on two cross-project pairs and ties on three cross-project pairs compared with LPDA_noDA.

On Promise dataset, the improvement of our method over LPDA_TA, LPDA_DA, LPDA_noDA and PLDA_noLS on average is at least by 23.44%, 2.55%, 2.28% , 3.32% and at most by 34.37%, 5.68%, 5.45%, 3.66% with regard to F-measure, Balance, MCC, and AUC, respectively. The results of Wilcoxon signed-rank statistical test shows that LPDA achieves more than 50 improvements with reference to F-measure, more than 32 improvements with reference to Balance, more than 35 improvements with reference to MCC and more than 43 improvements with reference to AUC respectively. As can be seen from the results shown in TableVII

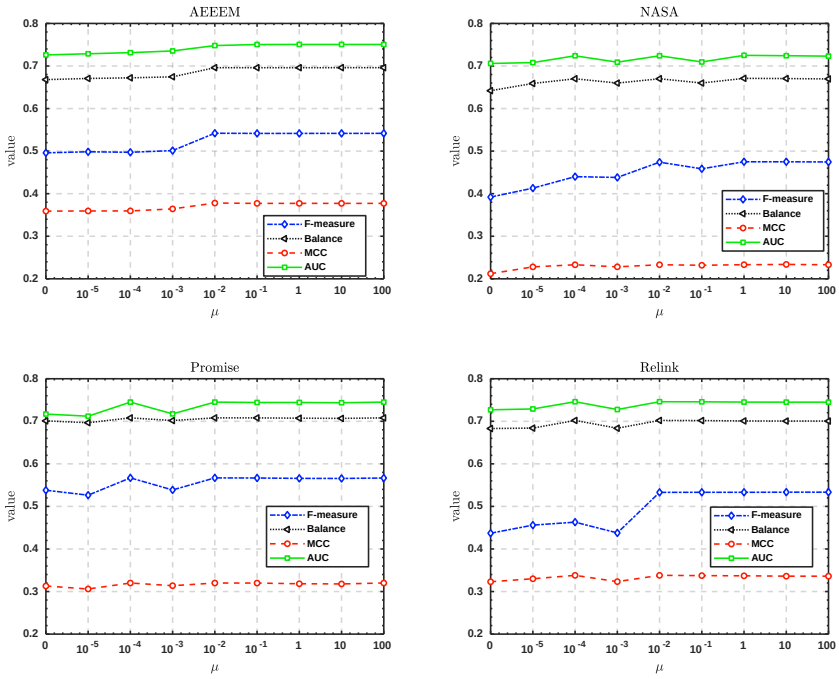


Fig. 7. Average f-measure, balance, MCC and AUC of LPDA on different datasets using differ trade-off parameter μ.

that LPDA attains more than 85 improvements in term of F-measure , more than 70 improvements with reference to Balance, more than 63 improvements with reference to MCC , and more than 68 improvements with reference to AUC, respectively.

VI. Discussion

A. Parameters Sensitivity

We explored the parameter sensitivity of LPDA to confirm that a diverse range of parameter values can be utilized to achieve a satisfactory level of performance. There are three parameters in LPDA, including the regularization parameter λ, the trade-off parameters μ and η. We change the values of these parameters in the range of {0, 0.00001, 0001, 0.001, 0.01, 0.1, 1, 10,1000} to report the overall average F-measure, Balance, AUC, and MCC on different datasets at 30 random points of times. It is worth noticing when μ=0,LPDA becomes LPDA_noDA, which is without discriminant term, when η=0, LDPA becomes LDPA_noLP, which is without locality preserving term, and when μ=0 and η=0, LDPA becomes LPDA_TA.

Fig. 7 presents the aggregate average value of F-measure, Balance, MCC, and AUC for LPDA across various datasets, influenced by varying the trade-off parameter μ. This figure suggests that indicates that lower emphasis on discriminant distribution alignment (reflected by a smaller μ value) leads to less accurate LPDA predictions. Additionally, the g-measure exhibits steady fluctuation within the μ, parameter range of [0.01, 100]. We plotted the overall average F-measure, Balance, MCC and AUC of LPDA with different trade-off parameter η on four different datasets which are shown in Fig. 8. We find that LPDA is sensitive to η and robust to η in [0.001, 10].

We plotted the overall average F-measure, Balance, MCC and AUC of LPDA with different trade-off parameter λ on four different datasets which are shown in Fig. 9. We find LPDA is sensitive to λ and the robust results of λ on different datasets are different. We observe that $\lambda \in [0.001, 1]$ is the robust prediction result for AEEEM and NASA datasets. $\lambda \in [0.01, 1]$ and $\lambda \in [0.1, 100]$ are the robust parameter values for Promise and Relink datasets, respectively.

VII. Threats to Validity

A. Internal Validity

The primary challenges to internal validity stem from unmanaged variables affecting the experimental procedure. An instance of this is the emergence of defects during the reimplementation of baseline methods, which may occur during the coding phase. To mitigate these issues, especially for baselines with openly accessible source code (e.g., TCA and TPTL methods), we utilize the provided source code to minimize potential discrepancies. For baselines without accessible source code, we make every effort to ensure precise implementation by meticulously adhering to the instructions outlined in relevant research studies.

B. External Validity

External validity mainly concerns that our research can be generalized to other software projects. Twenty-two software projects are employed in our experiment. We evaluated our methods against ten CPDP approaches using five performance evaluation metrics. Additionally, we employed both the Wilcoxon signed-rank test and the Scott Knott-ESD test for further analysis. However, we still cannot guarantee the consistence of our method on other software projects not covered

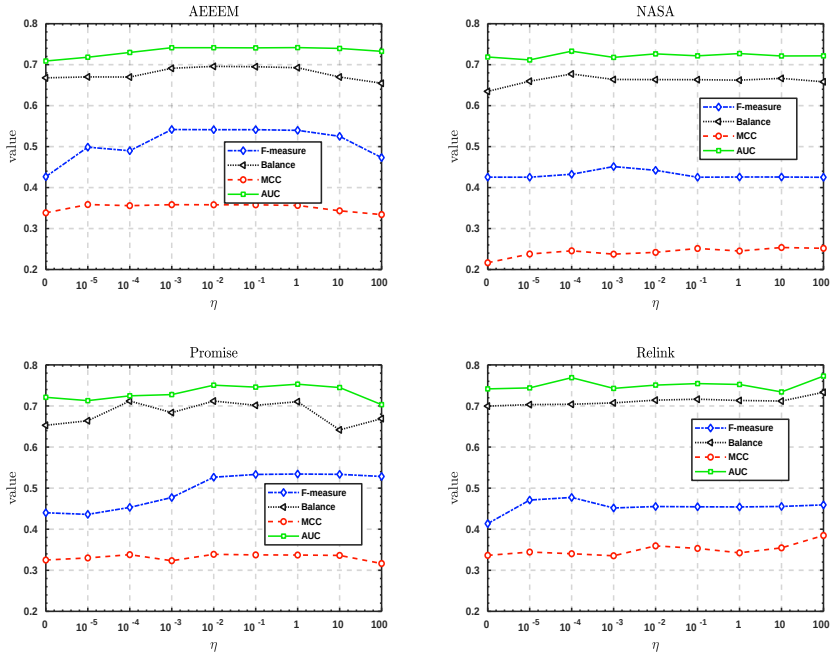


Fig. 8. Average f-measure, balance, MCC and AUC of LPDA on different datasets using differ trade-off parameter η.

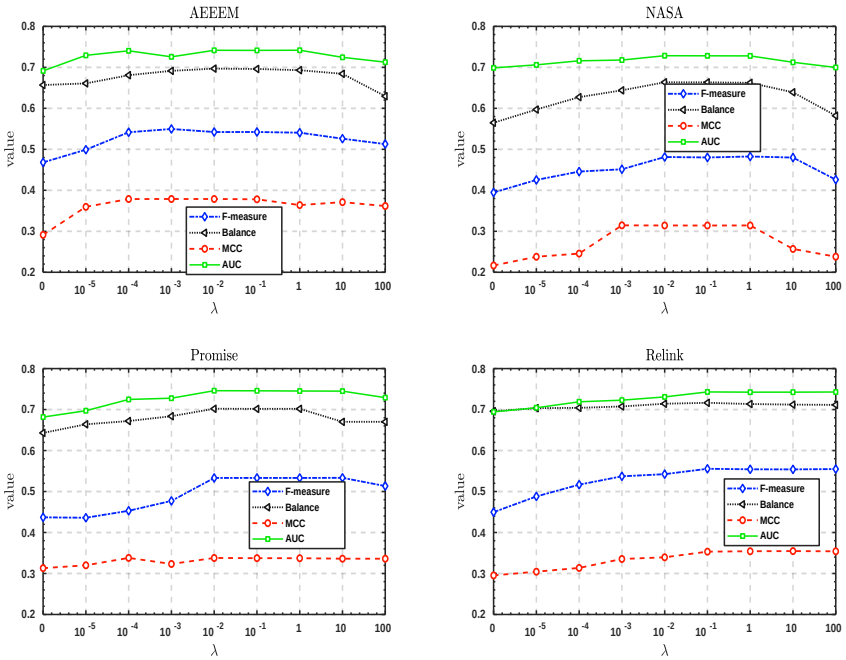


Fig. 9. Average f-measure, balance, MCC and AUC of LPDA on different datasets using differ trade-off parameter λ.

in this article. Further investigations on the applications of our research to commercial projects are needed.

VIII. Conclusion

In this paper, we propose a novel representation learning method LPDA for CPDP, which efficiently learned representations: 1) achieving transferability between two projects, and 2) preserving the local geometry to achieve discriminability between different categories. The local geometry is preserved by minimizing the distances between instances and their nearest neighbors within the same categories and maximizing the distances between instances and their neighbors in the different categories. The performance of LPDA is evaluated by five well-known evaluation indicators, including *Balance*, *F-measure*, *MCC* and *AUC*. We select ten state-of-the-art CPDP methods as baselines. We compare LPDA with ten baselines using the Wilcoxon signed-rank test and Scott-Knott ESD test (see Fig. 6). The experimental results on 22 software projects from four software repositories have shown that LPDA method is significantly superior to ten other CPDP methods. However, the proposed LPDA still has some limitations. LPDA does not consider important practical issues regarding the acquirement of useful knowledge from multiple external software projects and the application to projects. Moreover, LPDA method only considers traditional metric features and ignores effective high-level context features of source code. In our future work, we will further explore the potentials of LPDA.

Acknowledgment

The work was supported by the Guangdong Provincial Junior Innovative Talents Project for Regular Universities grant, No. 2023KQNCX005, and the Fundamental Research Funds for Central Universities.

References

[1] Li F, Yang P, Keung J W, et al. Revisiting 'revisiting supervised methods for effort-aware cross-project defect prediction'[J]. IET Software, 2023, 17(4): 472-495.

[2] Chang R, Mu X, Zhang L. Software defect prediction using non-negative matrix factorization[J]. Journal of Software, 2011, 6(11): 2114-2120.

[3] Zou Q, Lu L, Qiu S, et al. Correlation feature and instance weights transfer learning for cross project software defect prediction[J]. IET Software, 2021, 15(1): 55-74.

[4] Thwin M M T, Quah T S. Application of neural networks for software quality prediction using object-oriented metrics[J]. Journal of systems and software, 2005, 76(2): 147-156.

[5] Zou Q, Lu L, Yang Z, et al. Joint feature representation learning and progressive distribution matching for cross-project defect prediction[J]. Information and Software Technology, 2021, 137: 106588.

[6] Al-Laham M, Kassaymeh S, Al-Betar M A, et al. An efficient convergence-boosted salp swarm optimizer-based artificial neural network for the development of software fault prediction models[J]. Computers and Electrical Engineering, 2023, 111: 108923.

[7] Wu F, Jing X Y, Sun Y, et al. Cross-project and within-project semisupervised software defect prediction: A unified approach[J]. IEEE Transactions on Reliability, 2018, 67(2): 581-597.

[8] Jing X Y, Wu F, Dong X, et al. An improved SDA based defect prediction framework for both within-project and cross-project class-imbalance problems[J]. IEEE Transactions on Software Engineering, 2016, 43(4): 321-339.

[9] Zain Z M, Sakri S, Ismail N H A. Application of Deep Learning in Software Defect Prediction: Systematic Literature Review and Meta-analysis[J]. Information and Software Technology, 2023: 107175.

[10] Kanwar S, Awasthi L K, Shrivastava V. Candidate project selection in cross project defect prediction using hybrid method[J]. Expert Systems with Applications, 2023, 218: 119625.

[11] Ren Y, Liu B, Wang S. Joint Instance and Feature Adaptation for Heterogeneous Defect Prediction[J]. IEEE Transactions on Reliability, 2023.

[12] Sharma U, Sadam R. How far does the predictive decision impact the software project? The cost, service time, and failure analysis from a cross-project defect prediction model[J]. Journal of Systems and Software, 2023, 195: 111522.

[13] Xia X, Lo D, Pan S J, et al. HYDRRA: Massively compositional model for cross-project defect prediction[J]. IEEE Transactions on software Engineering, 2016, 42(10): 977-998.

[14] Yu Q, Jiang S, Qian J. Which is more important for cross-project defect prediction: instance or feature?[C]//2016 International Conference on Software Analysis, Testing and Evolution (SATE). IEEE, 2016: 90-95.

[15] Qiu S, Lu L, Jiang S. Joint distribution matching model for distribution–adaptation-based cross-project defect prediction[J]. IET Software, 2019, 13(5): 393-402.

[16] Liu C, Yang D, Xia X, et al. A two-phase transfer learning model for cross-project defect prediction[J]. Information and Software Technology, 2019, 107: 125-136.

[17] Jiang W, Qiu S, Liang T, et al. Cross-project clone consistent-defect prediction via transfer-learning method[J]. Information Sciences, 2023, 635: 138-150.

[18] Xu Z, Pang S, Zhang T, et al. Cross project defect prediction via balanced distribution adaptation based transfer learning[J]. Journal of Computer Science and Technology, 2019, 34: 1039-1062.

[19] Ma Y, Luo G, Zeng X, et al. Transfer learning for cross-company software defect prediction[J]. Information and Software Technology, 2012, 54(3): 248-256.

[20] Bennin K E, Keung J, Phannachitta P, et al. Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction[J]. IEEE Transactions on Software Engineering, 2017, 44(6): 534-550.

[21] Feng S, Keung J, Yu X, et al. COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction[J]. Information and Software Technology, 2021, 129: 106432.

[22] Bennin K E, Tahir A, MacDonell S G, et al. An empirical study on the effectiveness of data resampling approaches for cross-project software defect prediction[J]. IET Software, 2022, 16(2): 185-199.

[23] Gong L, Jiang S, Bo L, et al. A novel class-imbalance learning approach for both within-project and cross-project defect prediction[J]. IEEE Transactions on Reliability, 2019, 69(1): 40-54.

[24] Su W C, Huang C Y. Application of Weighted Combinations of Activation Functions to Defect Prediction in Software Development[J]. IEEE Transactions on Reliability, 2023.

[25] Bhat N A, Farooq S U. An empirical evaluation of defect prediction approaches in within-project and cross-project context[J]. Software Quality Journal, 2023: 1-30.

[26] Wang R, Nie F, Hong R, et al. Fast and orthogonal locality preserving projections for dimensionality reduction[J]. IEEE Transactions on Image Processing, 2017, 26(10): 5019-5030.

[27] Wang Q, Breckon T P. Cross-domain structure preserving projection for heterogeneous domain adaptation[J]. Pattern Recognition, 2022, 123: 108362.

[28] Qiang W, Li J, Zheng C, et al. Robust local preserving and global aligning network for adversarial domain adaptation[J]. IEEE Transactions on Knowledge and Data Engineering, 2021.

[29] Liu J, Lei J, Liao Z, et al. Software defect prediction model based on improved twin support vector machines[J]. Soft Computing, 2023: 1-10.

[30] Zhou Y, Yang Y, Lu H, et al. How far we have progressed in the journey? an examination of cross-project defect prediction[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2018, 27(1): 1-51.

[31] Zimmermann T, Nagappan N, Gall H, et al. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process[C]//Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. 2009: 91-100.

[32] Li Z, Jing X Y, Wu F, et al. Cost-sensitive transfer kernel canonical correlation analysis for heterogeneous defect prediction[J]. Automated Software Engineering, 2018, 25: 201-245.

[33] Turhan B, Menzies T, Bener A B, et al. On the relative value of cross-company and within-company data for defect prediction[J]. Empirical Software Engineering, 2009, 14: 540-578.

[34] Peters F, Menzies T, Marcus A. Better cross company defect prediction[C]//2013 10th Working Conference on Mining Software Repositories (MSR). IEEE, 2013: 409-418.

[35] Kawata K, Amasaki S, Yokogawa T. Improving relevancy filter methods for cross-project defect prediction[C]//2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence. IEEE, 2015: 2-7.

[36] Bhat N A, Farooq S U. An improved method for training data selection for cross-project defect prediction[J]. Arabian Journal for Science and Engineering, 2022: 1-16.

[37] Hosseini S, Turhan B, Mäntylä M. A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction[J]. Information and Software Technology, 2018, 95: 296-312.

[38] Pan S J, Tsang I W, Kwok J T, et al. Domain adaptation via transfer component analysis[J]. IEEE transactions on neural networks, 2010, 22(2): 199-210.

[39] Liu C, Yang D, Xia X, et al. A two-phase transfer learning model for cross-project defect prediction[J]. Information and Software Technology, 2019, 107: 125-136.

[40] Wang S, Liu T, Tan L. Automatically learning semantic features for defect prediction[C]//Proceedings of the 38th International Conference on Software Engineering. 2016: 297-308.

[41] Li J, He P, Zhu J, et al. Software defect prediction via convolutional neural network[C]//2017 IEEE international conference on software quality, reliability and security (QRS). IEEE, 2017: 318-328.

[42] Long M, Wang J, Ding G, et al. Transfer feature learning with joint distribution adaptation[C]//Proceedings of the IEEE international conference on computer vision. 2013: 2200-2207.

[43] Wang J, Chen Y, Hao S, et al. Balanced distribution adaptation for transfer learning[C]//2017 IEEE international conference on data mining (ICDM). IEEE, 2017: 1129-1134.

[44] Wang J, Feng W, Chen Y, et al. Visual domain adaptation with manifold embedded distribution alignment[C]//Proceedings of the 26th ACM international conference on Multimedia. 2018: 402-410.

[45] Zhang W, Wu D. Discriminative joint probability maximum mean discrepancy (DJP-MMD) for domain adaptation[C]//2020 international joint conference on neural networks (IJCNN). IEEE, 2020: 1-8.

[46] Wu H, Wu Q, Ng M K. Knowledge preserving and distribution alignment for heterogeneous domain adaptation[J]. ACM Transactions on Information Systems (TOIS), 2021, 40(1): 1-29.

[47] D'Ambros M, Lanza M, Robbes R. Evaluating defect prediction approaches: a benchmark and an extensive comparison[J]. Empirical Software Engineering, 2012, 17: 531-577.

[48] Siers M J, Islam M Z. Novel algorithms for cost-sensitive classification and knowledge discovery in class imbalanced datasets with an application to NASA software defects[J]. Information Sciences, 2018, 459: 53-70.

[49] Wu R, Zhang H, Kim S, et al. Relink: recovering links between bugs and changes[C]//Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering. 2011: 15-25.

[50] Jureczko M, Madeyski L. Towards identifying software project clusters with regard to defect prediction[C]//Proceedings of the 6th international conference on predictive models in software engineering. 2010: 1-10.

[51] Shao Y, Liu B, Wang S, et al. A novel software defect prediction based on atomic class-association rule mining[J]. Expert Systems with Applications, 2018, 114: 237-254.

[52] Shao Y, Liu B, Wang S, et al. Software defect prediction based on correlation weighted class association rule mining[J]. Knowledge-Based Systems, 2020, 196: 105742.

[53] Tantithamthavorn C, McIntosh S, Hassan A E, et al. An empirical comparison of model validation techniques for defect prediction models[J]. IEEE Transactions on Software Engineering, 2016, 43(1): 1-18.