Applied Intelligence
https://doi.org/10.1007/s10489-024-05459-1

Improving transfer learning for software cross-project defect prediction

Osayande P. Omondiagbe[1,2] · **Sherlock A. Licorish**[2] · **Stephen G. MacDonell**[3]

Accepted: 7 April 2024
© The Author(s) 2024

Abstract

Software cross-project defect prediction (CPDP) makes use of cross-project (CP) data to overcome the lack of data necessary to train well-performing software defect prediction (SDP) classifiers in the early stage of new software projects. Since the CP data (known as the source) may be different from the new project's data (known as the target), this makes it difficult for CPDP classifiers to perform well. In particular, it is a mismatch of data distributions between source and target that creates this difficulty. Transfer learning-based CPDP classifiers are designed to minimize these distribution differences. The first Transfer learning-based CPDP classifiers treated these differences equally, thereby degrading prediction performance. To this end, recent research has the Weighted Balanced Distribution Adaptation (W-BDA) method to leverage the importance of both distribution differences to improve classification performance. Although W-BDA has been shown to improve model performance in CPDP and tackle the class imbalance by balancing the class proportion of each domain, research to date has failed to consider model performance in light of increasing target data. We provide the first investigation studying the effects of increasing the target data when leveraging the importance of both distribution differences. We extend the initial W-BDA method and call this extension the W-BDA$^+$ method. To evaluate the effectiveness of W-BDA$^+$ for improving CPDP performance, we conduct eight experiments on 18 projects from four datasets, where data sampling was performed with different sampling methods. Data sampling was only performed on the baseline methods and not on our proposed W-BDA$^+$ and the original W-BDA because data sampling issues do not exist for these two methods. We evaluate our method using four complementary indicators (i.e., Balanced Accuracy, AUC, F-measure and G-Measure). Our findings reveal an average improvement of 6%, 7.5%, 10% and 12% for these four indicators when W-BDA$^+$ is compared to the original W-BDA and five other baseline methods (for all four of the sampling methods used). Also, as the target to source ratio is increased with different sampling methods, we observe a decrease in performance for the original W-BDA, with our W-BDA$^+$ approach outperforming the original W-BDA in most cases. Our results highlight the importance of having an awareness of the effect of the increasing availability of target data in CPDP scenarios when using a method that can handle the class imbalance problem.

Keywords Transfer learning · Cross-project defect prediction · Weighted balance distribution

1 Introduction

Software quality assurance (SQA) focuses on improving software development processes and preventing defects in production systems [33]. Given the limited budget and time available for such initiatives, SQA remains a challenging endeavor. Accurate software defect prediction (SDP) helps to minimise the time and effort necessary for testing software products by automating the process of detecting the areas of software that are prone to defects. In this paper, we use the term "defect" to refer to a fault or bug in software code. SDP relies on the building models on a training set. These training sets are usually labelled data consisting of module features. In most cases, the defect class contain less number compared to the non-defect class, thereby leading to imbalance class. The module features are typically extracted from the software history data taken from the teams' software repository [44]. The model is then used to predict the defect status labels of unlabeled modules within the same software project. This type of software prediction is usually termed within-project defect prediction (WPDP). Although recent work has explored how early in the software life cycle it is useful to use within-

✉ Osayande P. Omondiagbe
omondiagbep@landcareresearch.co.nz

Extended author information available on the last page of the article

Published online: 24 April 2024

project data without relying on cross-project data [41], these approaches may not be useful in instances where there is little data for the software project in question. Given this potential dilemma, various works have proposed a technique called cross-project defect prediction (CPDP) [40], which aims to utilise datasets from some external projects (termed a source project) to conduct defect prediction. This is usually done by using few datasets from the target project (i.e., the project under consideration and where there is a need to identify defects before software is released). Since different software module often yields different defect in the source and target, it is possible that data distribution differences across projects will be evident in this scenario.

Standard CPDPs are not designed to solve these distribution differences because the machine learning models employed were designed based on an assumption that the training and testing data subsets used in model construction follow the same distribution [48]. Accordingly, CPDP can result in poor model performance, as shown in the literature [1]. This indicated that (i) CPDP is a serious challenge, and (ii) the reason for the poor performance was due to the data distribution differences. To mitigate this data distribution difference, transfer learning-based CPDP methods were designed, focusing on distribution adaptation to reduce the distribution divergence between domains [22]. The two types of distribution difference addressed are marginal and conditional differences [45]. The marginal distribution difference refers to the probability distribution of the various values of the variables (e.g., software module features) in the source domain without reference to the target domain. The conditional distribution difference refers to the probability of the class label given the observations in either the source or target domain [52].

Previous transfer learning-based CPDP methods focus on addressing the marginal distribution [54], conditional distribution [2] or both [29, 52] (e.g., Balanced Distribution Adaptation (BDA)). There it is shown that when both distributions are adapted the model's performance increases [29]. However, these distribution differences are usually treated equally [56], and the importance of each individual distribution is not taken into consideration. This could be a detriment to the overall model design because when the data of the two projects (source and target) are dissimilar, the marginal distribution is more important than the conditional distribution, and when the two projects are very similar, the conditional distribution is more important [46].

Beyond the treatment of marginal and conditional distributions, the work by Cheng et al [7] shows that when more target data is available, transfer learning-based models are not able to extract information from the sources in a way that is not conflicting with the target data. This means negative transfer[1] is more likely to occur [17]. Similarly, the work of Ganchev et al [15] shows a performance increase when the source data is larger compared to the target data. This improvement decreases as the target data size increases. They noted an exception to this trend that occurs with 10% of the target data; in such case, no benefit was derived from using transfer learning because of the small target data (i.e all transferred rules were discarded). Although the dataset used in Ganchev et al [15] experiments are different from CPDP, it has been shown that machine learning robustness and generalisation are fundamentally correlated [51]. Also, If the ratio of the target and source domain is not taken into consideration, then as more target data is added to the learning system as the target data grows, the contribution of the source data in the system will be gradually shifted to fine-tuning the model [48], thereby leading to negative transfer. Furthermore, the inclusion of additional target data in the learning system may result in a decline in performance due to data shift. However, the Weighted Balanced Distribution Adaptation (W-BDA) method only tackles this issue by considering differences in marginal and conditional distributions [45], which can lead to negative transfer. We acknowledge that there are indeed alternative techniques that could potentially improve the model's performance in CPDP. However, this paper primarily focuses on transfer learning methods for addressing distribution differences. **Therefore, we are motivated to design a learning method that effectively addresses both distribution discrepancies and class imbalance while preventing negative transfer.** One example of applying this approach is using historical defect data from previous projects (e.g. web applications) to potentially gain valuable insights for predicting defects in a new project (e.g. e-commerce platform) and avoiding negative transfer in the process.

For this reason, we extend the W-BDA method [45], which works by adjusting the weight of each domain class while adaptively adjusting the importance of both the marginal and conditional distributions. The decision to extend the existing W-BDA model instead of developing a completely new approach was based on several factors.

Firstly, the W-BDA model has shown promising results in CPDP performance by addressing data distribution discrepancies and class imbalance [52]. It has been established as a strong baseline method in the field. Secondly, by extending the W-BDA model, we aimed to build upon its strengths and further enhance its capabilities. Extending an existing model provides several benefits:

[1] An interference of the previous knowledge with new learning, where one set of events could hurt the overall learning process

1. Practicality and adoption: Extending an existing model can facilitate the practical implementation and adoption of the proposed approach. Researchers and practitioners who are already familiar with the W-BDA model can easily adopt and integrate the extended version into their existing frameworks and workflows. This promotes the practical applicability and dissemination of the research findings.
2. Building on existing knowledge: The W-BDA model has been thoroughly studied and its performance evaluated in previous research. By extending this model, we can leverage the insights gained from those studies and build upon the established foundations. This saves time and effort by not starting from scratch and allows us to benefit from the accumulated knowledge.
3. Incremental improvements: Extending an existing model allows for incremental improvements, focusing on specific aspects that can enhance its performance. By identifying limitations or areas for improvement in the original W-BDA model, we can design extensions that address these issues more effectively. This iterative approach enables us to fine-tune and refine the model's capabilities.
4. Compatibility and comparability: Extending an existing model ensures compatibility and comparability with previous studies. Researchers can directly compare the performance of the extended model with the original W-BDA and other baseline methods, providing a clearer understanding of the improvements achieved.

This decision was driven by the aim to leverage the strengths of the W-BDA model while addressing its limitations and advancing the field of cross-project defect prediction. In summary, extending the existing W-BDA model allows us to build on established knowledge, make incremental improvements, ensure compatibility and comparability with previous studies, and enhance the practical adoption of the proposed approach. As this method does not take into account the amount of target data when adapting the distribution differences and handling the class imbalance between the source and target, the performance could be degraded as more target data become available due to negative transfer. To avoid such a problem, the intuition here is to incorporate prior knowledge of the ratio of the target and source data in the learning process when simultaneously addressing the distribution differences.

We call our method Weighted Balanced Distribution Adaptation plus (W-BDA$^+$). We simulate a CPDP environment by using our proposed method (i.e., W-BDA$^+$) and other CPDP methods to tackle both the marginal and conditional distribution difference in the presence/absence of class imbalance. We use the following indicators for evaluation; (i) Balanced Accuracy (BACC), (ii) Area Under the Receiver Operating Characteristic curve (AUC), (iii) F-measure (F1) and (iv) G-Measure (GM). These indicators were chosen because they were recently recommended for class imbalance learning [47] and have been used in defect prediction studies [23, 32, 43]. In fact, these indicators were also used in a related study, which is use here for comparison [52]. In this paper we place greater emphasis on the BACC because it takes into account the class imbalances and also overcomes bias in binary cases [8].

Our contributions in this work are two-fold.

One: We introduce a modified W-BDA method for CPDP called W-BDA$^+$. W-BDA simultaneously adjusting the weight of each class while addressing both marginal and conditional distributions. This was done by adaptively adjusting the importance of both distributions. The weight of each class is adjusted by considering the importance of the two types of distribution differences with a balance factor. We incorporate the ratio of target and source data in the learning process for extending the existing W-BDA.

Two: To validate the performance of our approach, we experiment with 5 other transfer learning CPDP methods, including two simpler methods (Bruka Filter [55] and Hybrid Instance Selection using Nearest Neighbour (HISNN) [38]) and three complex methods (Transfer Component Analysis (TCA) [36], Joint Distribution Adaptation (JDA) [29], Weighted Balanced Distribution Adaptation (W-BDA) [45]). The selection of the five baseline methods was based on the diversity of approaches, their usage in previous research, and the range of complexity they represent. In this work, we addressed class imbalance in our four baseline methods (JDA, Bruka, HISNN and TCA) and use the original dataset where class imbalance is present in our proposed approach (W-BDA$^+$) and the original W-BDA. This is because, W-BDA handles the class imbalance problem. For the four baseline methods, we addressed class imbalance using four standard methods (ADAYSN [5], Smote [6], Random under-sampling [28] and Random oversampling [4]). When compared to other approaches, experimental results show that our adapted W-BDA$^+$ is superior to all other CPDP methods.

Overall, our findings provide new insights that could help advance the CPDP field. For reproducibility and to aid further research, all experimental code and data are publicly available[2].

The remainder of the paper is organised as follows. First, we introduce related CPDP work and develop our research questions in Section 2. Next, we describe the technical details of the W-BDA and how we adapted the method in Section 3. In Section 4, we present our benchmark datasets and experimental design. We then present our experimental results in Section 5. We discuss our results and their implications in

[2] https://github.com/pascal082/cpdp-applied-intelligence-journal

Section 6. In Section 7, we outline potential threats to the validity of our outcomes. Finally, this paper concludes and identifies avenues for future work in Section 8.

2 Related work

Transfer learning-based approaches have been seen in the literature to address the data distribution differences. They are designed to transform both the source and target software module data into a new feature space. Once this transformation is done, the distributions of both the source and target data become similar. With this approach, no source data are discarded. [34] proposed an extension of the Transfer Component Analysis (TCA) [36] method called Transfer Component Analysis plus (TCA+). This method uses rules to find the best strategy to normalize the data before learning specific transfer components across domains in a Reproducing Kernel Hilbert Space (RKHS). In their RKHS learning, they used the Maximum Mean Discrepancy (MMD) to measure the distance. Their method was applied to two projects and eight defect datasets. The results show an increase in performance as compared to the original TCA method. The methods mentioned above only address the marginal distribution difference and neglect the conditional distribution differences. Joint Distribution Adaptation (JDA) [29] was designed to reduce the marginal and conditional distribution differences jointly with equal weights between the source and target projects. This method was experimented with Xu et al [52] on 18 projects from four datasets. The result indicates that the JDA approach performed better than other methods that only address marginal distribution differences.

Other methods such as transfer cost-sensitive boosting "TCSBoost" [39] and hybrid instance selection using nearest neighbour "HISNN" [38] have considered both transfer knowledge and class imbalance simultaneously. The TCSBoost calculates the similarity weight between the source and the target data, then uses a re-sampling method to balance the data distribution of both the defect and non-defect class of the source project. A cost-sensitive boost method is applied to address the distribution differences between the source and target projects, while the HISNN removes outliers in the source data that might hinder prediction performance. Thereafter, a Naive Bayes algorithm is used to classify instances in the target data. [39] indicated that the TCSBoost method could handle cases where there is a high degree of data imbalance.

In understanding the importance of both the marginal and conditional distributions in a transfer learning setting, the work of Long et al [29] was designed to understand how best to treat these two distribution differences and the way these differences impact overall model performance. They found that by treating the distribution differences separately or jointly but equally, they were able to construct new feature representation that is effective and robust for handling the distribution difference. However, it was established that greater discrepancy between distributions could lead to reduced model performance [29]. The Balanced Distribution Adaptation (BDA) was thus developed by Wang et al [45] to address this concern. BDA works by adapting both the marginal and conditional distributions between domains, and also leveraging the importance of those two distributions. Extensive experiments were conducted in cross-software defect prediction settings where BDA was seen to outperform other filter and transfer learning-based methods [52]. By treating the classes as balanced across the source and target domain when dealing with an imbalanced dataset, this could result in performance degradation [53]. Based on BDA, Wang et al [45] proposed a novel Weighted Balanced Distribution Adaptation (W-BDA) algorithm to tackle the class imbalance issue in transfer learning. The proposed W-BDA can adaptively change the weight of each class when performing distribution adaptation.

Methods such as Transfer Naive Bayes (TNB) and TCA+ showed improved performance compared to other methods, but they only addressed marginal distribution differences. JDA aimed to reduce both marginal and conditional distribution differences with equal weights between the source and target projects. However, some methods, including TCSBoost and HISNN, considered transfer knowledge and class imbalance simultaneously.

The Balanced Distribution Adaptation (BDA) and Weighted Balanced Distribution Adaptation (W-BDA) methods were developed to address distribution differences and class imbalance. W-BDA introduced adaptive weighting for each class during distribution adaptation. However, both the BDA and W-BDA do not take into account the ratio of the source and target data during learning. As noted from previous work where the performance decreases as the target data size increases [15], we extend the existing W-BDA to incorporate the proportion of the target and source data as prior knowledge in the learning stage, towards extending the body of work on transfer learning for defect prediction. We also justify this new constant that accounts for the source and target ratio in Section 3. To guide our investigation and evaluate the effectiveness of incorporating the ratio of the target and source data as prior knowledge in a transfer CPDP setting, we pose the following two research questions (RQs).

RQ1. How does the modelling performance of an approach that incorporates source and target ratios, leveraging both marginal and conditional distribution differences, compare to other transfer learning-based methods in CPDP?

Motivation: The distribution differences of CPDP data are often derived from marginal and conditional distribution differences [52]. Though W-BDA was proposed to address both distribution differences, this method does not consider

the fact that, when more target data become available, the model is forced to accept more source data, leading to performance degradation. Taking this issue into consideration, this research question is designed to investigate whether the proposed W-BDA$^+$ is better than the transfer-based and filter-based methods in a CPDP setting.

RQ2. What is the effect of incorporating source and target ratios in a model that considers both marginal and conditional distribution differences to address domain class imbalance issues in transfer learning?

Motivation: Though W-BDA was designed to address both distribution differences (marginal and conditional) simultaneously [52], its performance in a data imbalanced setting where there is limited target and abundant source data needs to be further explored. This question is designed to investigate whether and how W-BDA$^+$ (an adaptation of W-BDA to accommodate the ratios of the source and target data during learning) is superior to the initial W-BDA methods in a CPDP setting, and how. W-BDA$^+$ especially explores model performance gains when dealing with domain class imbalance in the presence of existing target data.

3 Background Knowledge

Here we introduce key concepts relating to sofwtare cross-project defect prediction (CPDP).

3.1 Source and target domain

In CPDP, the source domain refers to a dataset or software project from which we have labelled data and knowledge about defects. On the other hand, the target domain represents a different dataset or software project for which we want to predict defects using the knowledge acquired from the source domain [52, 57]. For example, let us consider an example where we have a source domain dataset containing defect information from a previous software project A, and our objective is to predict defects in a new target project B. In this scenario, project A serves as the source domain, providing labelled data that we can use to train our predictive models. Project B represents the target domain, where we aim to apply the knowledge and insights gained from project A to predict defects in the new context.

3.2 Marginal distribution

Marginal distribution refers to the distribution of a single variable or feature in a dataset, without considering any other variables [9]. In the context of defect prediction, the marginal distribution can represent the frequency of defects in a software module. For instance, if we have a dataset that contains defect information for multiple modules, analysing the marginal distribution would provide insights into the overall defect rate across all modules.

3.3 Conditional distribution

The conditional distribution describes the probability distribution of one variable given knowledge of another variable [9]. In the context of defect prediction, conditional distribution helps us understand the likelihood of a defect occurring in a software module based on specific features or characteristics of that module. Let us consider a scenario where our objective is to predict the presence of defects in software modules using two features: lines of code (LOC) and complexity. In this case, the conditional distribution would provide valuable insights into the probability of a defect given the values of LOC and complexity. By analyzing the conditional distribution, we can gain an understanding of how these features influence the likelihood of defect occurrence in the modules. This understanding allows us to quantify the relationship between the features and the presence of defects, enabling us to identify patterns and make accurate predictions based on this knowledge.

3.4 Data distribution discrepancies

Data distribution discrepancies refer to the differences or inconsistencies between the distributions of data from different domains or sources [52]. In the context of CPDP, data distribution discrepancies relate to variations in defect patterns or characteristics between different software projects or environments. For instance, let us consider a scenario where we have a source domain dataset comprising defect information from one software project, and a target domain dataset from a different project. If we observe noteworthy differences in defect patterns, such as the types of defects or their frequency, between the two projects, it indicates the presence of data distribution discrepancies. These discrepancies necessitate careful attention when transferring knowledge from the source to the target domain to ensure accurate and dependable defect predictions. Gaining a thorough understanding of data distribution discrepancies and effectively managing them is pivotal for achieving effective CPDP [52]. By acknowledging and addressing these variations, we can develop robust models that proficiently capture the underlying patterns and characteristics of defects in diverse software projects or environments.

3.5 Class imbalance

Class imbalance occurs when a dataset's distribution of classes, such as defective versus non-defective modules, is heavily skewed, with one class being significantly more prevalent than the other [30]. In defect prediction, this often

arises because the number of non-defective modules is typically much higher than the number of defective modules [30]. For example, let us consider a dataset of 1,000 software modules, where only 100 modules are defective. This creates a class imbalance problem as the majority class (non-defective modules) heavily outweighs the minority class (defective modules). Class imbalance can pose challenges for accurate defect prediction models, as they tend to become biased towards the majority class and may overlook the minority class [30]. Addressing class imbalance is crucial to develop reliable defect prediction models [30]. Techniques such as oversampling the minority class [30], undersampling the majority class [16], or using advanced algorithms specifically designed for imbalanced data can help mitigate the impact of class imbalance [31]. By appropriately handling class imbalance, we can ensure that the predictive models give due consideration to both defective and non-defective modules.

4 Method

4.1 Cross software defect prediction problem definition

The issue in play with cross-software defect prediction is seen in a labelled software module source dataset $\{\mathbf{x}_{s_i}, y_{s_i}\}_{i=1}^{n}$, an unlabeled software module target dataset $\{\mathbf{x}_{t_j}\}_{j=1}^{m}$, an assumed feature space $\mathcal{X}_s = \mathcal{X}_t$, label space $\mathcal{Y}_s = \mathcal{Y}_t$, marginal distributions $P_s(\mathbf{x}_s) \neq P_t(\mathbf{x}_t)$, with conditional distributions $P_s(y_s|\mathbf{x}_s) \neq P_t(y_t|\mathbf{x}_t)$. Transfer learning will aim to learn the labels $\mathbf{y}_t$ of $\mathcal{D}_t$ using the source domain $\mathcal{D}_s$. Weighted balanced distribution adaptation solves the transfer learning problem by adaptively minimizing the marginal and conditional distribution difference between domains. It also handle s the class imbalance problem. This is done to minimize the discrepancies between the marginal distribution $P(\mathbf{x}_s)$ and $P(\mathbf{x}_t)$, and conditional distribution $P(y_s|\mathbf{x}_s)$ and $P(y_t|\mathbf{x}_t)$.

4.2 Weighted balanced distribution adaptation

In the CPDP setting, transfer learning aims to adapt both the marginal and conditional distributions between domains [42]. This is done to minimize the distance between $\mathcal{D}_s$ and $\mathcal{D}_t$ as shown in (1).

$$D(\mathcal{D}_s, \mathcal{D}_t) \approx D(P(\mathbf{x}_s), P(\mathbf{x}_t)) + D(P(y_s|\mathbf{x}_s), P(y_t|\mathbf{x}_t)) \tag{1}$$

However, by matching both distributions and assuming they are equally important, it does not always hold. BDA and W-BDA are proposed to adaptively adjust the importance of both the marginal and conditional distributions based on the task to solve. It is worth nothing that BDA exploits a balance factor μ to leverage the different importance of distributions as shown in (2).

$$D(\mathcal{D}_s, \mathcal{D}_t) \approx (1-\mu) D(P(\mathbf{x}_s), P(\mathbf{x}_t)) + \mu D(P(y_s|\mathbf{x}_s), P(y_t|\mathbf{x}_t)) \tag{2}$$

In (2) $\mu \in [0, 1]$ is a balance factor which is used to leverage the distribution differences. When μ moves to 0, this indicates the CPDP datasets are more dissimilar, hence the marginal distribution should be paid more attention. On the other hand, as $\mu \to 1$, this shows the CPDP datasets are very similar, so the conditional distribution should be given more attention. By using the balance factor μ, the method can adaptively leverage the importance of each distribution in the CPDP dataset which could lead to improved performance.

It is also important to note that, since the target project $\mathcal{D}_t$ has no labels, it is feasible to evaluate the conditional distribution $P(y_t|\mathbf{x}_t)$. To evaluate the conditional distribution $P(y_t|\mathbf{x}_t)$, the class conditional distribution $P(\mathbf{x}_t|y_t)$ is used to approximate the conditional distribution $P(y_t|\mathbf{x}_t)$. To calculate the divergences between two marginal and two conditional distributions in (2), the maximum mean discrepancy (MMD) [36] is used. This is shown in (3). Since MMD is a non-parametric measurement, MMD has been widely applied to many existing transfer learning approaches [36].

$$D(\mathcal{D}_s, \mathcal{D}_t) \approx (1-\mu) \left\| \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_{s_i} - \frac{1}{m}\sum_{j=1}^{m} \mathbf{x}_{t_j} \right\|_{\mathcal{H}}^2 + \mu \sum_{c=1}^{C} \left\| \frac{1}{n_c} \sum_{\mathbf{x}_{s_i} \in \mathcal{D}_s^{(c)}} \mathbf{x}_{s_i} - \frac{1}{m_c} \sum_{\mathbf{x}_{t_j} \in \mathcal{D}_t^{(c)}} \mathbf{x}_{t_j} \right\|_{\mathcal{H}}^2 \tag{3}$$

In (3) $\mathcal{H}$H denotes the reproducing kernel Hilbert space (RKHS), c is the different class label which is 2 in CPDP, and n and m denote the number of software modules in the source and target data. The $\mathcal{D}_s^{(c)}$ and $\mathcal{D}_t^{(c)}$ denote the software module belonging to class c in source and target domain, respectively. The $n_c = |\mathcal{D}_s^{(c)}|, m_c = |\mathcal{D}_t^{(c)}|$, denoting the number of samples belonging to $\mathcal{D}_s^{(c)}$ and $\mathcal{D}_t^{(c)}$ respectively. The first and second terms in (3) denote the marginal distribution and conditional distribution distance between domains.

By further taking advantage of matrix regularization, (3) may be formalized as (4).

$$\begin{aligned} &\min\ \operatorname{tr}\left(\mathbf{A}^{\top}\mathbf{X}\left((1-\mu)\mathbf{M}_0+\mu\sum_{c=1}^{C}\mathbf{M}_c\right)\mathbf{X}^{\top}\mathbf{A}\right)+\lambda\|\mathbf{A}\|_F^2 \\ &\text{s.t. } \mathbf{A}^{\top}\mathbf{X}\mathbf{H}\mathbf{X}^{\top}\mathbf{A}=\mathbf{I}, \quad 0\le\mu\le 1 \end{aligned} \tag{4}$$

In (4) $\mathbf{X}$ denotes the input data matrix which is made of $\mathbf{x}_s$ and $\mathbf{x}_t$, $\mathbf{A}$ denotes the transformation matrix, $\mathbf{I}\in\mathbb{R}^{(n+m)\times(n+m)}$ represents the identity matrix, and $\mathbf{H}=\mathbf{I}-(1/n)\mathbf{1}$ is the centring matrix. Similar to the work of [36], $\mathbf{M}_0$ and $\mathbf{M}_c$ are the MMD matrices and can be calculated using (5) and (6) below. The first term in (4) represents a balance factor μ, while λ is the regularization parameter where $\|\cdot\|_F^2$ is the Frobenius norm. Two constraints are involved in (4): the first constraint ensures that the transformed data ($\mathbf{A}^{\top}\mathbf{X}$) should preserve the inner properties of the original data, and the second constraint enforces the range of the balance factor μ.

$$(\mathbf{M}_0)_{ij}=\begin{cases}\frac{1}{n^2}, & \mathbf{x}_i,\mathbf{x}_j\in\mathcal{D}_s\\ \frac{1}{m^2}, & \mathbf{x}_i,\mathbf{x}_j\in\mathcal{D}_t\\ -\frac{1}{mn}, & \text{otherwise}\end{cases} \tag{5}$$

$$(\mathbf{M}_c)_{ij}=\begin{cases}\frac{1}{n_c^2}, & \mathbf{x}_i,\mathbf{x}_j\in\mathcal{D}_s^{(c)}\\ \frac{1}{m_c^2}, & \mathbf{x}_i,\mathbf{x}_j\in\mathcal{D}_t^{(c)}\\ -\frac{1}{m_c n_c}, & \begin{cases}\mathbf{x}_i\in\mathcal{D}_s^{(c)},\mathbf{x}_j\in\mathcal{D}_t^{(c)}\\ \mathbf{x}_i\in\mathcal{D}_t^{(c)},\mathbf{x}_j\in\mathcal{D}_s^{(c)}\end{cases}\\ 0, & \text{otherwise}\end{cases} \tag{6}$$

To solve (4), we define the Lagrange multipliers as $\Phi=(\phi_1,\phi_2,\cdots,\phi_d)$. (4) is then re-written as:

$$\begin{aligned} L=\ &\operatorname{tr}\left(\mathbf{A}^{\top}\mathbf{X}\left((1-\mu)\mathbf{M}_0+\mu\sum_{c=1}^{C}\mathbf{M}_c\right)\mathbf{X}^{\top}\mathbf{A}\right)\\ &+\lambda\|\mathbf{A}\|_F^2+\operatorname{tr}\left((\mathbf{I}-\mathbf{A}^{\top}\mathbf{X}\mathbf{H}\mathbf{X}^{\top}\mathbf{A})\Phi\right) \end{aligned} \tag{7}$$

Next, the first-order $\partial L/\partial\mathbf{A}=0$ derivatives of (7) are set to 0 and the optimization is transformed into a generalized eigen vector composition problem as shown in the (8) below:

$$\begin{aligned} &\left(\mathbf{X}\left((1-\mu)\mathbf{M}_0+\mu\sum_{c=1}^{C}\mathbf{M}_c\right)\mathbf{X}^{\top}+\lambda\mathbf{I}\right)\mathbf{A}\\ &\qquad=\mathbf{X}\mathbf{H}\mathbf{X}^{\top}\mathbf{A}\Phi \end{aligned} \tag{8}$$

The optimal transformation matrix $\mathbf{A}$ is obtained by solving (8) and finding its $\mathbf{d}$ smallest eigen vectors. To address the conditional distribution for a class imbalance problem seen in CPDP, a more robust approximation is needed to utilize the class conditional distributions, i.e. to approximate $P(y|\mathbf{x})$. The proposed approximation by [45] is shown in (9) below.

$$\begin{aligned} &\|P(y_s|\mathbf{x}_s)-P(y_t|\mathbf{x}_t)\|_{\mathcal{H}}^2\\ &=\left\|\frac{P(y_s)}{P(\mathbf{x}_s)}P(\mathbf{x}_s|y_s)-\frac{P(y_t)}{P(\mathbf{x}_t)}P(\mathbf{x}_t|y_t)\right\|_{\mathcal{H}}^2\\ &=\|\alpha_s P(\mathbf{x}_s|y_s)-\alpha_t P(\mathbf{x}_t|y_t)\|_{\mathcal{H}}^2 \end{aligned} \tag{9}$$

In (9), α_s and α_t are approximated by the class proportion of each class in both software projects. The weighted balanced distribution adaptation (W-BDA) is proposed to balance the class proportion of each class in the software projects. Then, a weight matrix $\mathbf{W}_c$ for each class is constructed as shown below:

$$(\mathbf{W}_c)_{ij}=\begin{cases}\frac{P\left(y_s^{(c)}\right)}{n_c^2}, & \mathbf{x}_i,\mathbf{x}_j\in\mathcal{D}_s^{(c)}\\ \frac{P\left(y_t^{(c)}\right)}{m_c^2}, & \mathbf{x}_i,\mathbf{x}_j\in\mathcal{D}_t^{(c)}\\ -\frac{\sqrt{P\left(y_s^{(c)}\right)P\left(y_t^{(c)}\right)}}{m_c n_c}, & \begin{cases}\mathbf{x}_i\in\mathcal{D}_s^{(c)},\mathbf{x}_j\in\mathcal{D}_t^{(c)}\\ \mathbf{x}_i\in\mathcal{D}_t^{(c)},\mathbf{x}_j\in\mathcal{D}_s^{(c)}\end{cases}\\ 0, & \text{otherwise}\end{cases} \tag{10}$$

where $P\left(y_s^{(c)}\right)$ and $P\left(y_t^{(c)}\right)$ denote the class probability in the source and target domain, respectively. Embedding (10) into (9), we get the trace optimization problem of W-BDA:

$$\begin{aligned} &\min\ \operatorname{tr}\left(\mathbf{A}^{\top}\mathbf{X}\left((1-\mu)\mathbf{M}_0+\mu\sum_{c=1}^{C}\mathbf{W}_c\right)\mathbf{X}^{\top}\mathbf{A}\right)+\lambda\|\mathbf{A}\|_F^2\\ &\text{s.t. } \mathbf{A}^{\top}\mathbf{X}\mathbf{H}\mathbf{X}^{\top}\mathbf{A}=\mathbf{I}, \quad 0\le\mu\le 1 \end{aligned} \tag{11}$$

Here, we adjust for W-BDA^{+} by setting $\mu=\mu+\epsilon$, where ϵ = (**nt/ ns** *100), where nt = number of software module instance in the target dataset, and ns = number of software module instance in the source dataset. The balance factor is modified in (11) which considers the class probability and provides a more accurate approximation to the conditional distributions when handling the class imbalance. Since the class imbalance is affected by both the source and target datasets, the balance factor is modified to incorporate the proportion of target and source data by adding a new factor, which is denoted as ϵ. Given that both the source and target

dataset could have different distribution, and this distribution difference is widened with additional data [48], the new introduced factor(ϵ) helps to further change the weight of each class based on the source/target data when performing distribution adaptation. A final classifier is used to classify the module as defective or non-defective once the adaptation learning process is completed. We present the W-BDA$^+$ algorithm in Algorithm 1 and an overview of our W-BDA+ is shown in Fig 1.

Algorithm 1 W-BDA^+: Weighted Balanced Distribution Adaptation Plus

Input:

- Source feature matrix: $\mathbf{X}_s$
- Target feature matrix: $\mathbf{X}_t$
- Source label vector: $\mathbf{y}_s$
- dimension d
- balance factor μ
- regularization parameter λ
- number of target data **nt**
- number of source data **ns**

Output: Transformation matrix $\mathbf{A}$ and final classifier f

- Train a base classifier on $\mathbf{X}_s$ and apply prediction on $\mathbf{X}_t$ to get its soft labels $\hat{\mathbf{y}}_t$. Construct $\mathbf{X} = [\mathbf{X}_s, \mathbf{X_t}]$, initialize $\mathbf{M}_0$ and $\mathbf{M}_c$ using (10)
  - **repeat**
    Solve the eigen vector composition problem in (11) and use d smallest eigen vectors to build $\mathbf{A}$
    Train a classifier f on $\{\mathbf{A}^\top \mathbf{X}_s, \mathbf{y}_s\}$
    Update the soft labels of $\mathcal{D}_t$: $\hat{\mathbf{y}}_t = f(\mathbf{A}^\top \mathbf{X}_t)$
    Update matrix $\mathbf{M}_c$ using using (10) for W-BDA
  - **until** Convergence
  - **return** Classifier f

The algorithm time complexity using the Big O notation is $O(m(n1 + n2)^2)$, where $n1 + n2$ is the eigenvectors extracted, O is the order of magnitude and m is the leading eigenvalues. The time complexity analysis is based on the computational steps involved in solving the eigenvalue decomposition problem and computing the eigenvectors. The number of eigenvectors being computed is n1 + n2, and each eigenvector computation has a complexity of $O((n1 + n2)^2)$. Considering that there are m leading eigenvalues, the overall time complexity of the algorithm is $O(m(n1 + n2)^2)$. This time complexity analysis provides an understanding of how our proposed algorithm's performance scales with the size of the eigenvectors and the number of leading eigenvalues. It serves as a useful metric for evaluating the algorithm's efficiency and estimating its runtime on different datasets. Our method addresses the problem of performance degradation when more target data is added to the learning model by incorporating a weight adjustment mechanism in the adaptation process. The weights assigned to each class during adaptation are updated based on the ratio between the source and target data. This mechanism enables the method to dynamically adjust the contribution of each class during the adaptation, ensuring a balanced representation of the data and preventing the model from being biased towards either the source or target class distributions. By considering the source and target ratios, the W-BDA$^+$ method aims to strike a balance between the source and target domains, ensuring that both datasets contribute meaningfully to the learning process. This adaptive weighting approach helps to maintain model performance as more target data is incorporated, reducing the risk of performance degradation that can occur when class imbalances become more pronounced.

5 Experimental setup

5.1 Benchmark datasets

In this work, we conduct experiments on four software defect benchmark datasets, the AEEEM, NASA, SOFTLAB, and RELINK datasets. These datasets are described below:

- **AEEEM Dataset**: The AEEEM dataset was compiled by D'Ambros et al [11], and contains five open source projects with 5,371 data points. The projects include: Apache Lucene (LC), Equinox (EQ), Eclipse JDT Core (JDT), Eclipse PDE UI (PDE) and Mylyn (ML). The projects are Java-based, and are all uniform in terms of the number of features. The project data in the dataset comprise 61 different features (or columns/metrics), including 17 source code features, 5 previous-defect features, 5 entropy-of-change features, 17 entropy-of-source-code features, and 17 source code churn features. For more details on the metrics used for the source code features in Table 1, and how they are derived, please refer to [20]. In terms of the defect data, the metrics include the overall count of bugs and critical bugs found in the software projects.
- **NASA Dataset**: The publicly available NASA datasets have been studied extensively in software defect prediction work [2, 26, 35]. The projects are C-based and are extracted from a software system which is made of a set of static code features. This static code features include McCabe complexity, Halstead complexity, Cyclomatic complexity, number of lines, and other common features (i.e. features that are informative and useful in understanding software quality). In this paper, we use data for five out of 14 projects (CM1, MW1, PC1, PC3, and PC4) which share the same features as our earlier dataset. This was also used in the work of [52], where they experimented with the Balanced Distribution Adaptation (BDA) algorithm to solve cross project defect.

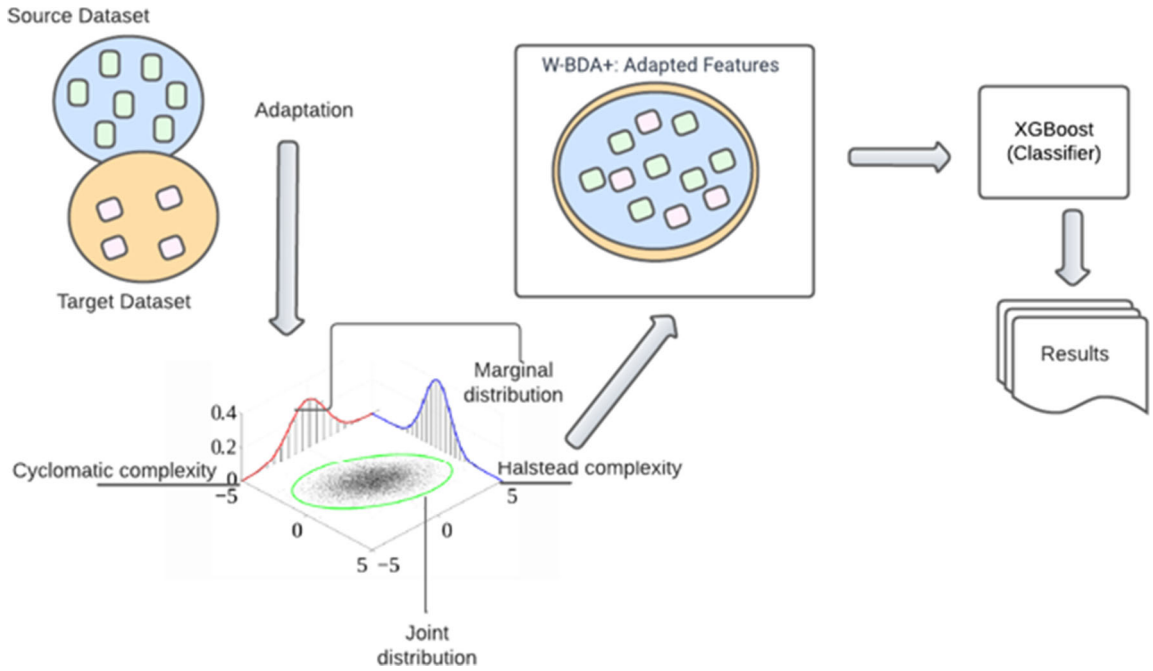


Fig. 1 An Overview of our proposed approach

- **SOFTLAB Dataset**: The five project data (ar1, ar3, ar4, ar5, and ar6) in this dataset were from a Turkish software company which develops embedded controllers for home appliances [21]. Each project data consist of 29 static code features.
- **RELINK Dataset**: This dataset [49] has 26 static code features and contains three projects (Apache HTTP Server (Apache), OpenIntents Safe (Safe), and ZXing). This dataset was used in previous defect prediction studies such as experimenting with the BDA algorithm [52] and Transfer Component Analysis (TCA) [34].

Table 1 Properties of the overall benchmark dataset

Dataset	Project	Description	# M	# DM	% DM	Class	Granularity	# F
AEEEM	EQ	OSGi framework	324	129	39.8	Java	Class	61
	JDT	IDE development	997	206	20.7			
	LC	Search Engine library	691	64	9.3			
	ML	Task management	1862	245	13.2			
	PDE	Java	1497	209	14.0			
SOFTLAB	ar1	Embedded controller for white-goods	121	9	7.4	C	Function	29
	ar3	Washing machine	63	8	12.7			
	ar4	Dish washer	107	20	18.7			
	ar5	Refrigerator	36	8	22.2			
	ar6	Embedded controller for white-goods	102	15	14.5			
NASA	CM1	Spacecraft instrument	327	42	12.8	C	Function	38
	MW1	A zero gravity experiment about combustion	251	25	10			
	PC1	Flight software for earth orbiting satellite	696	55	7.9			
	PC3	Flight software for earth orbiting satellite	1 073	132	12.3			
	PC5	Java	1 276	176	13.8			
RELINK	Apache	Web sever	194	98	50.5	Java	File	26
	Safe	Security	56	22	39.3			
	Zxing	Bar-code scanning library	399	118	29.6			

The description for each data in our benchmark datasets are presented in Table 1. Columns # M, # F, % DM and # DM represent the number of modules, the number of features, percentage of defective modules and the number of defective module respectively. Of the AEEEM dataset, Table 1 shows that project ML is the largest, comprising 1862 software modules without defects and 245 modules with defects. PDE is also noteworthy, comprising 1497 software modules without defects and 209 modules with defects. The other five datasets comprise $<$ 1000 modules without defects, however, even though EQ has 453 modules altogether, 129 of those were defects modules (or 39.8% defect density). Table 1 shows that the JDT project also has a relatively high defect density of 20.7%, comprising 997 software modules without defects and 206 modules with defects. For a detailed description of all the metrics in the AEEEM dataset please refer to [11]. In the NASA dataset, project CM1 and MW1 have fewer than 400 modules with 42 and 25 defect-related modules respectively, while the other three projects (PC1, PC3 and PC5) comprise of a higher data point with a defect density ratio of more than 10%. Of the RELINK dataset, Table 1 shows that project zxing is the largest, comprising 399 software modules without defects and 118 modules with defects. The other two projects Safe and Apache within the RELINK dataset had less data point but with an higher defect density of 50.5% and 39.3% respectively.

5.2 Data splitting and validation

The goal of this study is to evaluate the impact of incorporating the source and target ratios in W-BDA when dealing with domain class imbalance issues in a CPDP setting (RQ1). In addition, we set out to assess the performance of this W-BDA customization against other transfer-based CPDP methods (RQ2). To set the scope for answering the two research questions, our experimental procedure, which caters to the CPDP scenario, is described as follows. Based on the guidance of prior studies and in conforming to convention [31, 35], we normalise all the software metrics that are used in our training and test sets with a Z-score method, because these metrics are of different scales.

To ensure that results are reproducible, we randomly shuffled all data with a set seed before assigning the train and test sets. Since we are simulating a scenario (i.e., abundant source but fewer target datasets) where transfer-based CPDP can be used, we first split the target data into testing/training (30/70 split), then we combined the target training set with the source data to have abundant training datasets before developing the transfer-based CPDP model. This is because transfer-based learning algorithms seek to address distribution differences between the source and target data. This combined dataset is then split into testing/training (30/70). The splitting was done using a random sub-sampling method, which is also known as Monte Carlo cross-validation or multiple holds, to split the target data into train and test pairs. This approach randomly split the data into subsets, which is repeated (we did 10 repeats). This method was chosen as random sub-sampling is shown to be asymptotically consistent [3], which results in more pessimistic predictions of the test (target in our case) data when compared to typical cross-validation [50]. To eliminate any bias in designing our transfer learning scenario, we select a target project where the proportion of the modules on that project to the overall modules in the dataset is less than 25% (e.g, in the RELINK dataset, only the project "safe" had a proportion of less than 25% when compare to the total number of modules in that dataset). This is because as the target data grows, the contribution of the source data in the system will be gradually shifted to fine-tuning the transfer-based model [48].

5.3 Hyperparameter tuning

In this experiment, we need a final classifier to classify the module into defect or non-defect once the transfer learning is done with the learning process. We use the XGBoost as our final base classifier because it was seen to have high predictive power, as reported in a previous study [18].

As tuning both the transfer learning methods and the base classifier (XGBoost in our case) could be computationally infeasible with limited computation resources. For this reason we opt for a strategy to dynamically allocate resources to a set of random configurations, but use a successive halving approach [25] to stop poorly performing configurations. As similar work on CPDP has shown the positive effect of Bayesian optimisation method in selecting the parameter space [24], we use a bandit-based method that combines the benefits of both Bayesian optimization and bandit-based scheduling.

The bandit-based method used was the Bayesian Optimization HyperBand (BOHB) algorithm [12]. To aid reproduciblity, we implemented this algorithm using the Tune library [27]. Within the Tune framework, the implementation of the BOHB algorithm could be achieved by using the TuneBOHB search algorithm and the BOHB scheduler.

To tune the algorithms (transfer learning and XGBoost), we change their parameters, namely learning rate, n-estimator, subsample and max depth. As we are using Bayesian Optimisation, we need to set an Objective function to minimise. This is done by selecting a metric to be tracked during training. In our experiment, since our final base classifier selected was XGBoost, we used both the log loss and the error as our metrics, and set these to be our loss function to minimize. In using the Tune library, we configure the hyperparameter space, i.e., the range of each hyperparameter values. The configuration required to search for each algorithm is given in Table 2. For our balance factor (μ), the model is then trained

Table 2 Parameters used for transfer learning methods

Model	Name	Description	Range
Bruka Filter	k	The number of neighbours to each point (default=10)	[1, 100]
W-BDA/W-BDA+	lambda	Lambda value in equation (default=1)	[0.5,10]
	Balance factor μ	Balance factor in equation (default=1)	[0,1]
TCA/ JDA	kernel dimension	The type of kernel (default='linear')	'primal', 'linear', 'rbf', 'sam'
	dimension	The dimension after transforming (default=5)	[5, max(N source, N target)]
	lambda	Lambda value in equation (default=1)	[0.000001, 100]
	gamma	Kernel bandwidth for 'rbf' kernel (default=1)	[0.00001, 100]
HISNN	MinHamm	Minimum distance	[3,10]
	HissNNeighbours	K-nearest neighbour (default=5)	[5,10]
	k	The number of neighbours to each point (default=10)	[1, 100]
XGBoost (base classifier)	N_estimator	The number of boosting stages to perform	[10, 100]
	Learning rate	Number to use to shrink the tree learning_rate	[0.0001, 0.3]
	max_ depth	The maximum depth limits of node tree.	[10,100]
	min_child_weight	minimum sum of instance weight needed in a child.	[1, 2, 3]
	Subsample	The fraction of samples to be used to fit the learner	[0.5, 1.0]

and evaluated for each value in the grid shown in Table 2. The values of μ and lambda that yield the best performance for our algorithm were chosen as the optimal values. We perform different sets of experiments by selecting a target dataset from each project and using the remaining dataset from the project as the source dataset. To simulate a scenario where CPDP is applicable, we set criteria for the target experiment. This criterion was set to have a target-to-source ratio of $\leq$ 0.2. Based on this criterion, we perform 8 sets of experiments as follows:

- **AEEEM Dataset**: We performed three main experiments by rotating the target dataset with the EQ, JDT and LC projects. These three datasets have limited modules when compared to the ME and PDE projects. This simulates a CPDP setting.
- **NASA Dataset**: We performed two main experiments by rotating the target dataset with the CM1 and MW1 project. These two datasets were selected as they had limited modules when compared to the PC1, PC3 and PC5 projects in the NASA dataset.
- **SOFTLAB Dataset**: Similarly, we performed two main experiments by rotating the target dataset with the ar3 and ar5 projects. These two datasets were selected as they had limited modules when compared to the ar1, ar4 and ar6 projects in the SOFTLAB dataset.
- **RELINK Dataset**: We performed a single experiment by using the Safe project as the target. This dataset has fewer module when compared to the Apache and Zxing projects in the RELINK dataset.

In doing this, we aim to understand the effect of having more of the target dataset in our proposed method. To further understand how the models used in this experiment react to various configurations of class imbalances, we perform four different types of sampling. The models used in this experiment were Transfer Component Analysis [36], Joint Distribution Adaptation (JDA) [29], Bruka Filter [55], Hybrid Instance Selection using Nearest Neighbour (HISNN) [38], Weighted Balanced Distribution Adaptation (W-BDA) [45] and our adapted version of the W-BDA. These techniques were experimented with under various configurations of data imbalance. Further details on these models are given in Sections 2 and 4. In evaluating our models' performances, the experiment was run N number of times. This was done to reduce the margin of error and in following convention [37]. We first run the experiment 10 times and the mean and standard deviation for each model were recorded. Then, we determine the standard error and use a 95% certainty for all model to estimate the accuracy of the mean.

To investigate the effect of negative transfer on the existing W-BDA and our proposed W-BDA$^+$ method when the ratio of the target and source domain is not taken into consideration, we further performed a single experiment for each dataset by selecting the project with the highest number of module as the target dataset. We selected the ML, PC5 and Zxing project as the target dataset for the AEEEM, NASA and RELINK experiments respectively. In each of three experiments we start by using a 10% of the target dataset and then we gradually increased the size of the target dataset.

Next, we configured five different scenarios by using four different sampling methods to sample our source data. The sampling methods used were the synthetic minority over-sampling (SMOTE) algorithm [6], adaptive synthetic sampling (ADASYN) algorithm [19], random over-sampling and random under-sampling [14]. We then combined the sampled output with the target training data. In each of these scenarios, we again use the Monte Carlo cross-validation to split the combined (source and target training data) into training (70%) and testing (30%), and build 6 different models. This was repeated 10 times. We used the configuration space defined in Table 2 for each model to find the optimal model. The experiment setup is shown in Fig. 2. Our environment comprised of a 16-Core CPU with 16GB RAM, and the experiments took approximately one hour for each target data experiment. This time-frame includes tuning both the W-BDA+ and the base classifier (XGboost).

5.4 Evaluation measures

We execute the experiments on a cluster machine. All codes were written and executed using Python 3.7. To evaluate the prediction accuracy of our modelling approach, we compute: **Balanced accuracy (BACC):** BACC measures model performance, taking into account class imbalances, and it also overcomes bias in binary cases [8]. The balanced accuracy is computed as the average of the proportion of correct predictions for each class separately. **Area Under the Receiver Operating Characteristic curve (AUC):** AUC is the area under the curve between the true positive rate and false-positive rate [41]. **G-measure (GM):** This is the harmonic mean between Recall and the compliment of False Positive Rate measured [41]. **F-measure (F1):** This is used for evaluating a binary classification model based on the predictions made for the positive class [32]. We use the BACC, AUC, GM and F1 to evaluate the importance of incorporating the source and target ratios in W-BDA when dealing with domain class imbalance issues in a CPDP setting (RQ1) . Additionally, we assess the performance of this customised W-BDA approach compared to other CPDP transfer learning-based

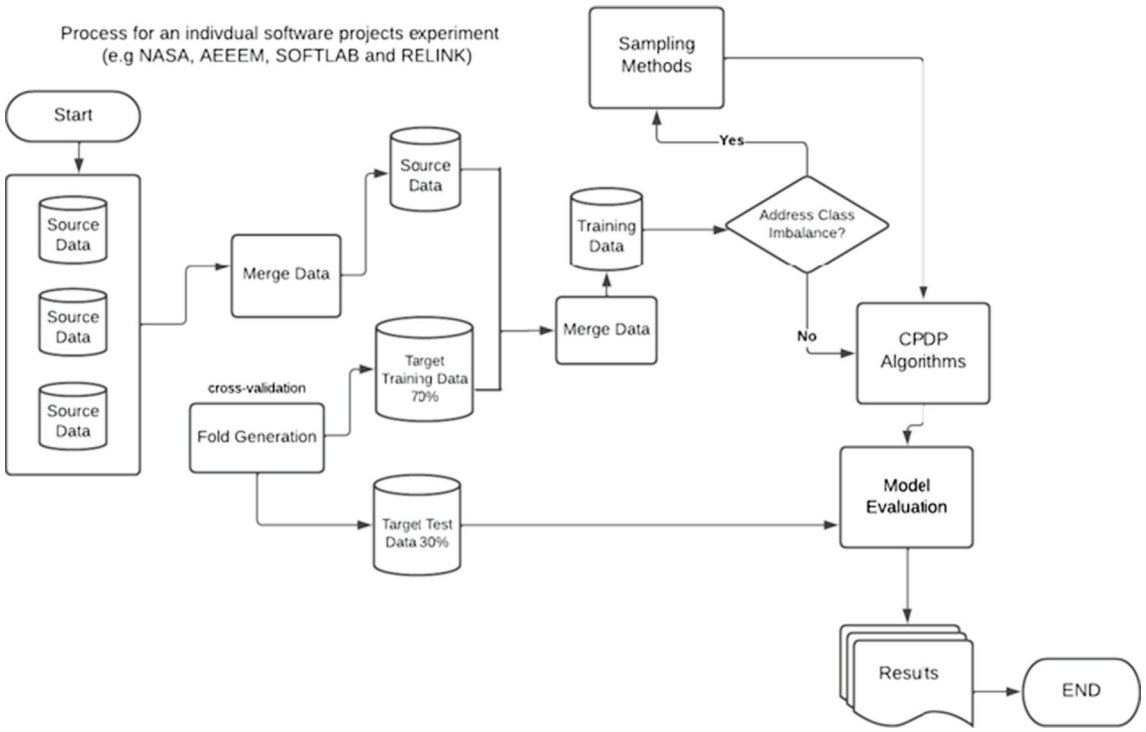


Fig. 2 An Overview of our data splitting, model construction and evaluation approach

methods highlighted in the related work section, which had been designed to address distribution differences (RQ2).

5.5 Statistical testing

For our statistical testing we place greater emphasis on the BACC because it takes into account the class imbalances and also overcomes bias in binary cases [8]. We compare distributions of BACC measures for various sampling approaches, which may have the same median but different distributions. To identify significant differences among these populations, we use the non-parametric Scott-Knott test [10], as recommended in previous studies and employed in prior software defect prediction research [41]. This test ranks classification models into distinct groups based on statistical significance (with an $\alpha = 0.05$). It follows a top-down bi-clustering approach, dividing the techniques into ranks based on the mean evaluation measure (BACC in this paper). Our 10 repeated measurements for each classification method are ranked based on the mean BACC. The Scott-Knott test iteratively divides the ranks only when the previously divided ranks are statistically significantly different. The test stops when further division into statistically distinct ranks is no longer possible. This approach overcomes the issue of overlapping groups encountered with other post hoc tests, such as Nemenyi's and Tukey HSD test [10].

6 Results

In this section, we present detailed experimental results for the indicators as proposed in Section 5 (BACC, AUC, GM and F1). We tuned the base and domain adaptation method in performing comparative analysis towards answering our research questions in turn.

RQ1: Table 3 - 6 report the average values of the four indicators for the five methods on the SOFTLAB, NASA, RELINK and AEEEM dataset. The " +" before each indicator in Table 3 -6 denotes the indicator that needs to be maximized (higher the better), while the "Overall Rank" column denotes the rank position found in the Scott-Knott rank. It shows that W-BDA$^+$ gets the best average performance on all indicators when compared with the four baseline methods. In particular, this was noted when no sampling method was used. More specifically, when compared with the four baseline, W-BDA$^+$ achieves improvement of 15% to 40% in terms of BACC, of 12% to 38% in terms of GM, of 10% to 42% in terms of AUC, and 9% to 34% in terms of F1 score.

Formal statistical testing was performed to compare the outcomes of the two models (W-BDA and W-BDA$^+$) under different sampling techniques. This test was carried out using Scott-Knott to rank the models, and the result is presented in the rank column. The statistical analysis was conducted for 5 models with 410 paired samples. For reproducibility, the family-wise significance level of the Scott-Knott tests was alpha=0.05. The results of the Scott-Knott test illustrate that

Table 3 AEEEM Datasets: CPDP methods vs W-BDA$^+$ with different sampling methods

Project	Target Dataset	CPDP Classifier - Sampling	Overall				
Rank	$BACC^+$	AUC^+	GM^+	$F1^+$			
AEEEM	EQ (324 sample)	JDA- ADASYN	9	0.771	0.786	0.76	0.76
		JDA- No Sampling	8	0.786	0.796	0.782	0.646
		JDA- Random Oversampling	13	0.743	0.756	0.736	0.646
		JDA- Random Undersampling	10	0.672	0.665	0.663	0.586
		JDA- Smote	10	0.741	0.753	0.746	0.653
		TCA- ADASYN	11	0.733	0.754	0.742	0.678
		TCA- No Sampling	14	0.659	0.699	0.663	0.598
		TCA- Random Oversampling	16	0.631	0.654	0.645	0.567
		TCA- Random Undersampling	15	0.638	0.702	0.645	0.547
		TCA- Smote	10	0.733	0.756	0.725	0.651
		HISNN- ADASYN	17	0.611	0.678	0.567	0.580
		HISNN- No Sampling	14	0.655	0.675	0.664	0.602
		HISNN- Random Oversampling	18	0.578	0.567	0.523	0.457
		HISNN- Random Undersampling	13	0.672	0.698	0.665	0.587
		HISNN- Smote	12	0.705	0.760	0.721	0.650
		Brukka- ADASYN	11	0.725	0.742	0.713	0.541
		Brukka - No Sampling	7	0.801	0.862	0.821	0.654
		Brukka- Random Oversampling	5	0.828	0.831	0.786	0.65
		Brukka- Random Undersampling	4	0.844	0.856	0.785	0.654
		Brukka- Smote	19	0.560	0.588	0.557	0.457
		W-BDA$^+$- No Sampling	1	0.964	0.978	0.968	0.879
	LC (691 sample)	JDA- ADASYN	8	0.647	0.679	0.634	0.548
		JDA- No Sampling	8	0.647	0.678	0.624	0.546
		JDA- Random Oversampling	9	0.6355	0.645	0.625	0.542
		JDA- Random Undersampling	8	0.648	0.678	0.665	0.579
		JDA- Smote	5	0.548	0.598	0.576	0.491
		TCA- ADASYN	8	0.575	0.574	0.527	0.435
		TCA- No Sampling	1	0.649	0.656	0.634	0.445
		TCA- Random Oversampling	3	0.557	0.587	0.554	0.465
		TCA- Random Undersampling	10	0.638	0.665	0.675	0.576
		TCA- Smote	5	0.591	0.612	0.609	0.578
		HISNN- ADASYN	10	0.623	0.657	0.634	0.563
		HISNN- No Sampling	8	0.535	0.558	0.524	0.476
		HISNN- Random Oversampling	11	0.506	0.554	0.523	0.498
		HISNN- Random Undersampling	10	0.562	0.574	0.571	0.435
		HISNN- Smote	11	0.571	0.587	0.564	0.502
		Brukka- ADASYN	6	0.726	0.779	0.695	0.674
		Brukka - No Sampling	6	0.735	0.756	0.749	0.624
		Brukka- Random Oversampling	5	0.728	0.757	0.737	0.692
		Brukka- Random Undersampling	4	0.734	0.752	0.696	0.646
		Brukka- Smote	19	0.506	0.578	0.524	0.676
		W-BDA$^+$- No Sampling	1	0.819	0.845	0.832	0.754

Table 3 continued

Project	Target Dataset	CPDP Classifier - Sampling	Overall				
	JDT (997 sample)	JDA- ADASYN	3	0.745	0.798	0.713	0.666
		JDA- No Sampling	12	0.565	0.587	0.559	0.456
		JDA- Random Oversampling	9	0.630	0.656	0.642	0.652
		JDA- Random Undersampling	8	0.664	0.668	0.635	0.598
		JDA- Smote	5	0.734	0.756	0.727	0.653
		TCA- ADASYN	8	0.659	0.696	0.646	0.587
		TCA- No Sampling	8	0.667	0.692	0.684	0.598
		TCA- Random Oversampling	3	0.747	0.798	0.712	0.616
		TCA- Random Undersampling	10	0.627	0.669	0.645	0.579
		TCA- Smote	5	0.737	0.798	0.719	0.652
		HISNN- ADASYN	10	0.627	0.642	0.634	0.556
		HISNN- No Sampling	8	0.646	0.672	0.632	0.576
		HISNN- Random Oversampling	11	0.578	0.597	0.587	0.476
		HISNN- Random Undersampling	12	0.566	0.596	0.582	0.509
		HISNN- Smote	11	0.582	0.604	0.575	0.521
		Brukka- ADASYN	6	0.725	0.746	0.719	0.634
		Brukka - No Sampling	6	0.722	0.746	0.739	0.652
		Brukka- Random Oversampling	5	0.732	0.743	0.745	0.687
		Brukka- Random Undersampling	10	0.628	0.652	0.643	0.587
		Brukka- Smote	7	0.696	0.734	0.707	0.627
		W-BDA$^+$- No Sampling	1	0.964	0.987	0.978	0.892

W-BDA$^+$ belongs to the top rank group and ranks the first or second on the BACC indicator. This was more noticeable when no sampling method was used in all experiments.

RQ2: To clearly understand the effect of incorporating the source and target ratios in W-BDA as proposed in algorithm 1 above, we ran the experiment for both the W-BDA and W-BDA$^+$ under different sampling techniques. Tables 7 - 10 report the average values of the four indicators for the two methods on the SOFTLAB, NASA, RELINK and AEEEM dataset. The "+" before each indicator in Tables 7 - 10 denote the indicator that needs to be maximized (higher the better), while the "Overall Rank" column denotes the rank position found in the Scott-Knott rank. It shows that W-BDA$^+$ gets the best average performance on all indicators when compared with the original WBDA for all eight experiments. In particular, this was noted when no sampling method was used. More specifically, when compared with the the original WBDA, W-BDA$^+$ achieves improvement of 7% to 13% in terms of BACC, of 7% to 14% in terms of GM, of 5% to 12% in terms of AUC, and 2% to 7% in terms of F1 score. Formal statistical testing was performed to compare the outcomes of the two models (W-BDA and W-BDA$^+$) under different sampling techniques. This test was carried out using Scott-Knott to rank the models with alpha=0.05. The statistical analysis was conducted for the two models with 20 paired samples, and the result is presented in the rank column. The results of the Scott-Knott test illustrate that W-BDA$^+$ with no sampling techniques belongs to the top rank group and ranks the first or second on the BACC indicator for all eight experiments.

7 Discussion and Implications

In this section, we discuss our findings considering our two research questions and their implications for research and practice in turn.

RQ1: Our results show that by incorporating the source and target ratios we were able to improve the existing W-BDA model by 4% when no sampling technique was applied, and approximately 7% with other sampling techniques when using the EQ and LC as targets. We also observed that by configuring the marginal and conditional distribution of the source and target data with equal weights (as recommended by others, e.g., [29]), or by adapting both the marginal and conditional distributions (as done in BDA), we were able to outperform methods that only address the marginal distribution differences (e.g., TCA, JDA), and methods based on selection (e.g., HISNN). Also, models that predict based on the similarity between the target and source data (Bruka model) tend to have a higher prediction accuracy than models which address the distribution differences (TCA, JDA). This pattern was seen when we used the EQ and LC datasets as

Table 4 SOFTLAB Datasets: CPDP methods vs W-BDA$^+$ with different sampling methods

Projects	Target Dataset	CPDP Classifier - Sampling	OverallRank	$BACC^+$	AUC^+	GM^+	$F1^+$
SOFTLAB	ar3 (63 sample)	JDA- ADASYN	6	0.550	0.576	0.554	0.398
		JDA- No Sampling	6	0.549	0.614	0.609	0.517
		JDA- Random Oversampling	7	0.538	0.576	0.547	0.457
		JDA- Random Undersampling	6	0.550	0.587	0.576	0.4066
		JDA- Smote	14	0.450	0.509	0.517	0.289
		TCA- ADASYN	10	0.477	0.498	0.480	0.356
		TCA- No Sampling	6	0.76	0.76	0.76	0.76
		TCA- Random Oversampling	13	0.460	0.520	0.498	0.365
		TCA- Random Undersampling	7	0.540	0.573	0.59	0.435
		TCA- Smote	9	0.550	0.576	0.587	0.498
		HISNN- ADASYN	8	0.536	0.556	0.519	0.432
		HISNN- No Sampling	15	0.438	0.447	0.442	0.398
		HISNN- Random Oversampling	16	0.408	0.445	0.418	0.276
		HISNN- Random Undersampling	12	0.464	0.486	0.498	0.376
		HISNN- Smote	11	0.474	0.489	0.487	0.352
		Brukka- ADASYN	4	0.627	0.654	0.632	0.542
		Brukka - No Sampling	1	0.638	0.675	0.643	0.557
		Brukka- Random Oversampling	3	0.630	0.654	0.644	0.576
		Brukka- Random Undersampling	2	0.637	0.666	0.645	0.597
		Brukka- Smote	12	0.465	0.487	0.477	0.452
		W-BDA$^+$- No Sampling	2	0.636	0.654	0.643	0.589
	ar5 (36 sample)	JDA- ADASYN	6	0.550	0.598	0.567	0.496
		JDA- No Sampling	6	0.549	0.598	0.536	0.432
		JDA- Random Oversampling	7	0.538	0.567	0.557	0.294
		JDA- Random Undersampling	6	0.550	0.598	0.568	0.492
		JDA- Smote	14	0.450	0.479	0.416	0.398
		TCA- ADASYN	10	0.477	0.498	0.476	0.398
		TCA- No Sampling	6	0.555	0.598	0.572	0.452
		TCA- Random Oversampling	13	0.460	0.498	0.475	0.376
		TCA- Random Undersampling	7	0.450	0.498	0.488	0.316
		TCA- Smote	9	0.494	0.547	0.526	0.466
		HISNN- ADASYN	8	0.536	0.572	0.555	0.453
		HISNN- No Sampling	15	0.4381	0.442	0.448	0.312
		HISNN- Random Oversampling	16	0.408	0.452	0.415	0.298
		HISNN- Random Undersampling	2	0.464	0.498	0.486	0.376
		HISNN- Smote	11	0.474	0.498	0.486	0.443
		Brukka- ADASYN	4	0.627	0.674	0.657	0.489
		Brukka - No Sampling	2	0.638	0.664	0.661	0.609
		Brukka- Random Oversampling	3	0.630	0.670	0.663	0.562
		Brukka- Random Undersampling	12	0.637	0.687	0.662	0.7598
		Brukka- Smote	12	0.463	0.492	0.447	0.362
		W-BDA$^+$- No Sampling	1	0.677	0.721	0.706	0.712

Table 5 NASA Datasets: CPDP methods vs W-BDA$^+$ with different sampling methods

Projects	Target Dataset	CPDP Classifier - Sampling	OverallRank	$BACC^+$	AUC^+	GM^+	$F1^+$
NASA	MW1 (251 sample)	JDA- ADASYN	7	0.610	0.665	0.642	0.598
		JDA- No Sampling	7	0.609	0.656	0.698	0.584
		JDA- Random Oversampling	7	0.598	0.642	0.623	0.512
		JDA- Random Undersampling	7	0.610	0.634	0.629	0.698
		JDA- Smote	15	0.510	0.553	0.545	0.520
		TCA- ADASYN	11	0.537	0.557	0.569	0.476
		TCA- No Sampling	8	0.660	0.698	0.616	0.592
		TCA- Random Oversampling	14	0.520	0.557	0.542	0.432
		TCA- Random Undersampling	8	0.600	0.653	0.661	0.598
		TCA- Smote	10	0.554	0.598	0.576	0.523
		HISNN- ADASYN	2	0.586	0.562	0.564	0.571
		HISNN- No Sampling	16	0.498	0.523	0.515	0.451
		HISNN- Random Oversampling	17	0.468	0.495	0.465	0.442
		HISNN- Random Undersampling	13	0.524	0.556	0.586	0.396
		HISNN- Smote	12	0.534	0.562	0.551	0.476
		Brukka- ADASYN	5	0.687	0.727	0.619	0.623
		Brukka - No Sampling	2	0.698	0.721	0.709	0.617
		Brukka- Random Oversampling	4	0.698	0.732	0.716	0.698
		Brukka- Random Under sampling	2	0.697	0.724	0.716	0.653
		Brukka- Smote	13	0.523	0.559	0.554	0.451
		W-BDA$^+$- No Sampling	2	0.736	0.798	0.7861	0.698
	CM1 (327 sample)	JDA- ADASYN	8	0.630	0.667	0.654	0.543
		JDA- No Sampling	8	0.629	0.665	0.643	0.545
		JDA- Random Oversampling	9	0.619	0.642	0.665	0.556
		JDA- Random Undersampling	8	0.630	0.656	0.643	0.598
		JDA- Smote	16	0.530	0.576	0.565	0.498
		TCA- ADASYN	12	0.557	0.587	0.597	0.464
		TCA- No Sampling	8	0.632	0.654	0.654	0.569
		TCA- Random Oversampling	15	0.540	0.532	0.554	0.432
		TCA- Random Undersampling	9	0.620	0.645	0.632	0.532
		TCA- Smote	11	0.574	0.587	0.552	0.432
		HISNN- ADASYN	10	0.606	0.654	0.641	0.587
		HISNN- No Sampling	17	0.518	0.554	0.576	0.476
		HISNN- Random Oversampling	18	0.488	0.521	0.554	0.476
		HISNN- Random Undersampling	14	0.544	0.563	0.543	0.454
		HISNN- Smote	13	0.554	0.576	0.598	0.456
		Brukka- ADASYN	6	0.707	0.745	0.734	0.617
		Brukka - No Sampling	4	0.718	0.754	0.736	0.676
		Brukka- Random Oversampling	5	0.710	0.756	0.729	0.654
		Brukka- Random Undersampling	4	0.717	0.743	0.778	0.698
		Brukka- Smote	14	0.543	0.598	0.562	0.463
		W-BDA$^+$- No Sampling	2	0.746	0.798	0.776	0.719

Table 6 RELINK Datasets: CPDP methods vs W-BDA$^+$ with different sampling methods

Projects	Target Dataset	CPDP Classifier - Sampling	OverallRank	$BACC^+$	AUC^+	GM^+	$F1^+$
RELINK	safe (56 sample)	JDA- ADASYN	6	0.540	0.576	0.557	0.444
		JDA- No Sampling	6	0.539	0.587	0.554	0.457
		JDA- Random Oversampling	7	0.528	0.557	0.576	0.510
		JDA- Random Undersampling	6	0.540	0.598	0.587	0.456
		JDA- Smote	13	0.440	0.459	0.476	0.396
		TCA- ADASYN	10	0.467	0.498	0.488	0.387
		TCA- No Sampling	6	0.542	0.559	0.556	0.498
		TCA- Random Oversampling	12	0.450	0.498	0.448	0.324
		TCA- Random Undersampling	7	0.530	0.542	0.510	0.398
		TCA- Smote	9	0.484	0.543	0.505	0.409
		HISNN- ADASYN	8	0.516	0.587	0.545	0.580
		HISNN- No Sampling	14	0.428	0.458	0.490	0.376
		HISNN- Random Oversampling	15	0.398	0.429	0.409	0.298
		HISNN- Random Undersampling	11	0.454	0.476	0.496	0.396
		HISNN- Smote	10	0.464	0.510	0.498	0.396
		Brukka- ADASYN	4	0.617	0.657	0.632	0.598
		Brukka - No Sampling	2	0.628	0.654	0.634	0.567
		Brukka- Random Oversampling	2	0.620	0.657	0.698	0.610
		Brukka- Random Undersampling	3	0.627	0.653	0.643	0.598
		Brukka- Smote	11	0.453	0.498	0.445	0.376
		W-BDA$^+$- No Sampling	1	0.696	0.719	0.758	0.698

Table 7 AEEEM Datasets: W-BDA vs W-BDA$^+$ with different sampling methods

Projects	Target Dataset	CPDP Classifier - Sampling	OverallRank	$BACC^+$	AUC^+	GM^+	$F1^+$
AEEEM	EQ (324 sample)	WBDA- No Sampling	2	0.919	0.934	0.907	0.823
		W-BDA$^+$- No Sampling	1	0.964	0.978	0.968	0.879
	LC (691 sample)	WBDA- No Sampling	2	0.776	0.795	0.734	0.679
		W-BDA$^+$- No Sampling	1	0.819	0.845	0.832	0.754
	JDT (997 sample)	WBDA- No Sampling	4	0.775	0.796	0.743	0.676
		W-BDA$^+$- No Sampling	1	0.964	0.987	0.978	0.892

Table 8 SOFTLAB Datasets: W-BDA vs W-BDA$^+$ with different sampling methods

Projects	Target Dataset	CPDP Classifier - Sampling	OverallRank	$BACC^+$	AUC^+	GM^+	$F1^+$
SOFTLAB	ar3 (63 sample)	WBDA- No Sampling	1	0.678	0.692	0.701	0.567
		W-BDA$^+$- No Sampling	2	0.636	0.654	0.643	0.589
	ar5 (36 sample)	WBDA- No Sampling	1	0.678	0.712	0.690	0.620
		W-BDA$^+$- No Sampling	1	0.677	0.721	0.706	0.712

Table 9 NASA Datasets: W-BDA vs W-BDA$^+$ with different sampling methods

Projects	Target Dataset	CPDP Classifier - Sampling	OverallRank	$BACC^+$	AUC^+	GM^+	$F1^+$
NASA	MW1 (251 sample)	WBDA- No Sampling	3	0.692	0.721	0.709	0.669
		W-BDA$^+$- No Sampling	1	0.736	0.798	0.7861	0.698
	CM1 (327 sample)	WBDA- No Sampling	2	0.702	0.743	0.676	0.654
		W-BDA$^+$- No Sampling	1	0.746	0.798	0.776	0.719

Table 10 RELINK Datasets: W-BDA vs W-BDA$^+$ with different sampling methods

Projects	Target Dataset	CPDP Classifier - Sampling	OverallRank	$BACC^+$	AUC^+	GM^+	$F1^+$
RELINK	safe (56 sample)	WBDA- No Sampling	1	0.698	0.736	0.712	0.699
		W-BDA$^+$- No Sampling	1	0.710	0.719	0.758	0.698

our target. A different observation was seen when we used the JDT data as the target dataset. In that experiment, the WBDA$^+$ model was not the best performing model. Further investigation needs to explore the causes of the patterns that were observed here.

Implications: We found that by addressing both marginal and conditional distributions, the overall performance of the model improved. The overall model performance becomes stronger when these two distribution differences are addressed adaptively, as seen in the W-BDA and our improved W-BDA$^+$ model when compared to the other model (TCA), which only addresses the marginal distribution. Also, when both distribution difference are addressed with adaptive weights (W-BDA) instead of weighting the distribution difference equally such as in the JDA method [29], we observed an 8% to 12% increase in performance for both the original W-BDA and our improved W-BDA$^+$ approach. In addressing the class imbalance before applying the various models, we found that the W-BDA outperformed "HISNN" [38], designed to consider both transfer knowledge and class imbalance simultaneously. Similar results were also seen in previous study [45], where they compare the first version of W-BDA called BDA with other transfer-based models (HISNN) and transfer filter-based models (Peter filter and NN-Filter). From our results, it is inconclusive to say that incorporating the source and target ratio in the W-BDA model will always be better because as we alternate the target data in datasets with more defect data and more number software modules, the results were different as shown in on our formal statistical testing. Filter-based methods are seen to be more effective in improving model prediction performance compared to models that address data distribution differences. This outcome indicates that practitioners developing models for such tasks should favour the choice of sample dataset to use as source than relying on addressing only one type of distribution difference. This is particularly necessary if practitioners are willing to leave the configuration of the marginal and conditional distributions to the techniques and not worry about these issues. Also, the sampling techniques used in a CPDP setting should be chosen carefully. The Scott-Knott rank shows our W-BDA$^+$ approach without any sampling technique was ranked first in all 8 experiments.

RQ2: From our experiment, we observed that when we incorporate the source and target ratios data into the W-BDA method, we reported a higher BACC score. This high BACC score was seen when the source data was not sampled in all three experiments. It is worth noting that in our experiments, we had a higher BACC when the target dataset was EQ and LC in all sampling techniques for the AEEEM project. Our outcomes here are similar to a previous work Xu et al [52], where they had reported that an earlier version of W-BDA called Balanced Distribution Adaptation (BDA) could improve model performance in CPDP. However, their work did not show how BDA performance varies with or without addressing class imbalance on the data level. We explored this further by using various sampling techniques on the W-BDA method and performed formal statistical testing where Section 6 shows the conditions under which the extension of W-BDA outperformed the original model. Again, the Scott-Knott rank shows our W-BDA$^+$ model ranked best in all 8 experiments except on two occasions (i.e., in SOFTLAB and RELINK project experiments), where we noticed the W-BDA was ranked in the same position with our W-BDA$^+$.

Implications: By incorporating the ratios of the target and source data in the W-BDA model, there is no need to address class imbalance at the data level with the use of sampling techniques. This is because the ratio compensated for changes in data imbalance, thereby ensuring optimum model performance.

By incorporating the ratios of target and source in the W-BDA model, we were able to tackle the domain class imbalance problem, and slightly improve the performance of the existing W-BDA model in a CPDP setting. The formal statistical tests in Section 6 indicated that the W-BDA$^+$ (the improved W-BDA) approach always ranked the first in all sampling techniques between the two models (W-BDA and W-BDA$^+$) when we used smaller target datasets such as the EQ and LC datasets (target to source ratio of 1:4). This suggests that the improved method will perform better without addressing the data imbalance issue from the data level. Although, W-BDA was proposed to address both distribution (marginal and conditional) differences using balance weighying, and was seen to achieve average improvements in a CPDP setting [52], it is clear that the ratio of the number of defects in the target and source could affect the overall modelling performance. These findings indicate that software engineering researchers who design or use models for CPDP tasks should be cautious to align specific sampling techniques with the model design to address both data marginal and conditional differences. The findings from this investigation

indicate that software practitioners designing or using CPDP models should decide carefully when to address class imbalance issues in the data level based on the type of model that is used.

Further analysis was carried out to understand the effect of not considering the source and target proportion in the W-BDA method. We performed three different experiments by using the ML, PC5 and Zxing project as the target dataset for the AEEEM, NASA and Relink experiments respectively. We tested the original W-BDA and our proposed W-BDA$^+$ method. This was done by keeping 10% of the target dataset as our test set and we gradually added a fraction of the remaining 90% of the target dataset into our training sets (see experiment process described in Fig 2 above). The results are shown in Fig 3 below where the blue line represents our W-BDA$^+$ and the red line represents the original W-BDA. It is evident that our proposed approach was able to prevent negative transfer. It is clearly shown that when the target data was increased, there was a decline in the performance of the original W-BDA. An opposite result was shown in our W-BDA$^+$. These findings highlight an important issue stated in previous work [48] (i.e., the contribution of the source data in the system will be gradually shifted to fine-tuning the model, thereby leading to negative transfer).

We formally restate and answer our research questions as follows:

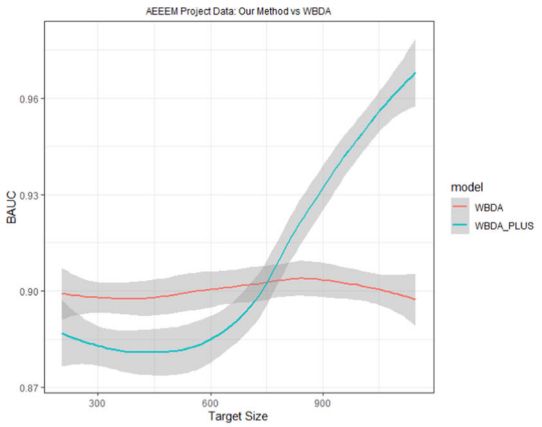
RQ1: How does the modelling performance of an approach that incorporates source and target ratios, leveraging both marginal and conditional distribution differences, compare to other transfer learning-based methods in CPDP?

Answer RQ1: Performance issues linked to the probability distribution differences between domains is one of the main challenges faced in transfer learning. In transfer-based CPDP, most existing transfer learning-based CPDP methods do not address both marginal and conditional distribution differences of data between different projects. A previously proposed model called Weighted-Balanced Distribution Adaptation (W-BDA), designed to adaptively handle the distribution difference and class imbalance, has ignored the fact that when more target data become available, adding more source data to the learning model could lead to performance degradation because it forces the data to be used for fine-tuning. Accordingly, we considered mitigating this problem by proposing an extension of the W-BDA, which we called W-BDA$^+$. Extensive experiments on four software defect benchmark datasets (AEEEM, SOFTLAB, RELINK and NASA) demonstrate the superiority of our method over the existing W-BDA and other transfer-based models that are typically used in a CPDP setting when data imbalance is not addressed.

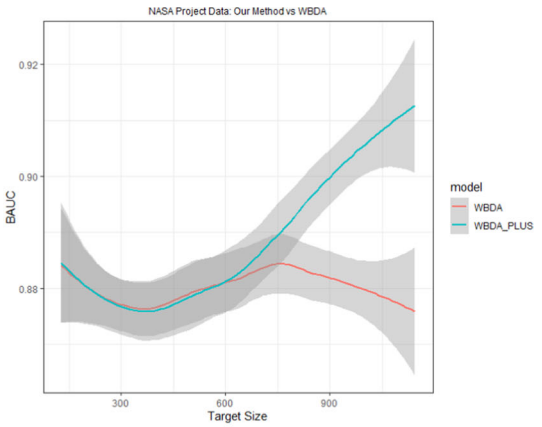
RQ2: What is the effect of incorporating source and target ratios in a model that considers both marginal and conditional distribution differences to address domain class imbalance issues in transfer learning?

Answer RQ2: By incorporating the ratios of the target and source in a method that can adaptively change the weight of each class when performing distribution adaptation, we are able to tackle the class imbalance problem and improve the performance of such an approach in a CPDP setting. This highlights the fact that when adaptively addressing distribution differences, we need to also consider the amount of target and source datasets in the learning system. By observing how an increase in the target datasets affects the adaptation process, it was evident that when the target data was increased, there was a decline in the performance of the original W-BDA. An opposite result was shown in our W-BDA$^+$. These findings indicate that the contribution of the source data in the system will be gradually shifted to fine-tuning as more target datasets are added to the learning system, thereby leading to negative transfer. It was evident that our proposed approach was able to prevent negative transfer.

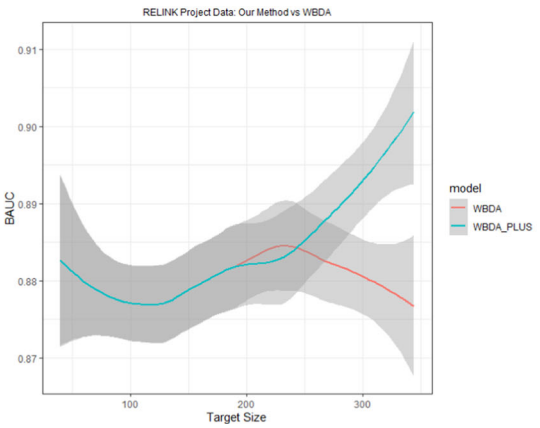
Interpretability: Since W-BDA$^+$ is a feature adaptation approach designed to enhance the performance of a baseline method such as XGBoost by addressing the challenges posed by class imbalance and distribution differences in transfer learning, it is important to highlight its interpretability. The use of feature importance can be adopted to explain its interpretability. The W-BDA$^+$ method can provide insights into the importance of features or variables contributing to the prediction of defects. By analysing the learned models or adaptation techniques, stakeholders can understand which factors have the most significant impact on defect prediction. This information would enable them to focus their efforts on improving or mitigating these critical areas. Additionally, the predictions generated by the W-BDA$^+$ method can be interpreted and explained to stakeholders. Instead of providing binary defect labels, the method can provide probabilistic predictions or confidence scores, indicating the likelihood of a particular code module containing a defect. Stakeholders can set appropriate thresholds based on their risk tolerance and make informed decisions regarding defect mitigation strategies. By emphasising the interpretability of the W-BDA$^+$ method, stakeholders can gain a comprehensive understanding of the factors influencing defect predictions. This understanding enables them to prioritize their efforts effectively, allocate resources efficiently, and take proactive measures to improve software quality. The interpretability of the method enhances decision-making, facilitates risk management, and empowers stakeholders to implement targeted defect mitigation strategies.



(a) AEEEM Project: W-BDA and W-BDA$^+$



(b) NASA Project: W-BDA and W-BDA$^+$



(c) RELINK Project: W-BDA and W-BDA$^+$

Fig. 3 Learning Curve

8 Threats to validity

We concede that there are a few uncontrolled factors that may have impacted the experimental results in this study. For instance, there could have been unexpected faults in the implementation of the methods [13]. We sought to reduce such threats by using the source code provided for the transfer learning methods that were used in this study (e.g., W-BDA, TCA, Bruka Filter, HISNN and JDA). These may be We were also concerned about threats relating to the CPDP techniques studied. To mitigate CPDP techniques threat when compared to the W-BDA method, we studied four main techniques recently found to be state of the art in transfer-based CPDP [24].

We also acknowledge the potential limitation associated with the choice of the model. In this paper, we have chosen to adopt XGBoost as the baseline model due to its high performance as demonstrated in a previous study [25]. We recognise that the choice of the baseline model can impact the results and the generalisability of our findings. Regarding threats due to overfitting, we took several precautions during the experimental design and evaluation process. Firstly, we selected a diverse range of benchmark datasets from various projects to ensure a comprehensive evaluation of the proposed method's performance. This selection aimed to minimize any bias towards specific datasets and increase the generalisability of our findings. Secondly, we followed standard practices in machine learning experimentation, including the use of cross-validation techniques, to reduce the risk of overfitting. By partitioning the data into training and validation sets, we were able to assess the model's performance on unseen data, providing a more reliable estimate of its generalisability. Overall, we acknowledge the potential concern of overfitting, and we have taken steps to mitigate this risk by employing diverse datasets, following standard evaluation practices, and planning for future evaluations on additional datasets.

Also, the limited number of software defect benchmark datasets used in our experiments poses a potential threat to the external validity of our findings. Although we aimed to select diverse datasets representing different domains, sizes, and characteristics commonly encountered in software

projects, these datasets may not fully encompass the entire range of software projects. As a result, the generalisability of our results to all possible software projects could be limited. However, these datasets have been widely used in the literature and are considered standard benchmarks for evaluating transfer learning approaches in the CPDP domain [24, 52]. The use of a small number of benchmark datasets also introduces a potential threat to construct validity. The performance and effectiveness of our proposed method may vary when applied to different datasets that were not included in our experiments. The specific characteristics and nuances of these excluded datasets could impact the results and potentially yield different conclusions. To mitigate these threats, we plan to address the external validity concern by incorporating additional benchmark datasets in future research.

Our proposed algorithm (W-BDA$^+$) may have limitations in various aspects. Firstly, it relies on addressing distribution differences between the source and target datasets. However, the effectiveness of this approach may vary depending on the specific characteristics of the distribution discrepancies. Different types of distribution discrepancies may require tailored adaptation techniques or alternative approaches. To overcome this limitation, further exploration could involve the use of more advanced adaptation methods or hybrid approaches that combine multiple strategies to effectively handle diverse distribution discrepancies. Secondly, our approach may have limitations in terms of scalability when applied to larger and more complex datasets. As the size and complexity of the data increase, the computational requirements and runtime of our method may pose a challenge. To overcome this limitation, it is important to explore techniques to enhance the efficiency and scalability of the proposed method. This may involve optimising algorithms, leveraging parallel computing, or investigating data reduction techniques to handle large-scale datasets more effectively, all good areas of future work.

Further enhancing our proposed method may involve hyperparameters that need to be carefully tuned to achieve optimal performance. The selection of those hyperparameters could impact the results and the generalisability of the approach. It is important to investigate and use methods for automated or data-driven hyperparameter selection and tuning to ensure robustness and reduce the dependency on manual tuning.

Finally, while we recognize the threats above, our study here contributes novel findings to transfer-based CPDP modelling.

9 Conclusion and Future Work

Addressing the probability distribution differences between domains is one of the main issues faced in transfer learning. In a transfer-based CPDP, most existing transfer learning-based CPDP methods do not address both marginal and conditional distribution differences of data between different projects. A previously proposed model called Weighted-Balanced Distribution Adaptation (W-BDA) designed to adaptively handle the distribution difference and class imbalance has ignored the fact that when more target data becomes available, adding more source data to the learning model could lead to performance degradation because it forces the data to be used for fine-tuning. Accordingly, we consider mitigating this problem by proposing an extension of the W-BDA, which we called W-BDA$^+$. Extensive experiments on four software defect benchmark datasets (AEEEM, SOFTLAB, RELINK and NASA) demonstrate the superiority of our methods over the existing W-BDA and other transfer-based models that are typically used in a CPDP setting when data imbalance is not addressed.

Our future work will focus on a large-scale study of the W-BDA$^+$ method on a wider range of software defect datasets. Our proposed method would also enhance other future work in the following areas. The W-BDA$^+$ method can be integrated into code review processes to support quality assurance efforts. Code review plays a crucial role in identifying defects, ensuring code quality, and maintaining software reliability. By leveraging knowledge from historical defect data, the method can help identify potential problematic code sections, guide code reviewers in focusing on critical areas, and so enhance the effectiveness of code review activities. In the maintenance phase of software projects or when dealing with legacy systems, the W-BDA$^+$ method can help identify potential defect-prone areas. By utilising historical defect data from similar projects or systems, maintenance teams can focus their efforts on critical areas, allocate resources accordingly, and proactively address potential issues. Additionally, the W-BDA$^+$ method can assist in test case prioritisation by identifying areas of the software system that are more likely to contain defects. This can aid in optimising testing efforts by focusing on critical areas, reducing the number of test cases needed, and improving the overall efficiency of the testing process. In agile software development environments, where frequent iterations and changes occur, the W-BDA$^+$ method can be utilised to leverage knowledge from previous projects to predict defects in new iterations. This can assist development teams in identifying potential areas of concern early on and allocating resources effectively for testing and bug fixing.

By exploring these practical applications, we aim to demonstrate the versatility and value of the W-BDA$^+$ method in real-world software engineering projects. Through its integration into code review, maintenance processes, and agile development practices, the method can contribute to improved defect detection, efficient resource allocation, and

enhanced software reliability. In addition, we hope to develop and package our solution for potential dashboard use.

Acknowledgements This research was partly supported by an Internal Research fund from Manaaki Whenua - Landcare Research, New Zealand. Special thanks to the Department of Informatics at Landcare Research for their ongoing support.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data and Code Availability The data and all experimental codes are publicly available and can be retrieved from https://github.com/pascal082/cpdp-applied-intelligence-journal. The code is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Amasaki S, Aman H, Yokogawa T (2022) An extended study on applicability and performance of homogeneous cross-project defect prediction approaches under homogeneous cross-company effort estimation situation. Empir Softw Eng 27(2):46
2. Bai J, Jia J, Capretz LF (2022) A three-stage transfer learning framework for multi-source cross-project software defect prediction. Inf Softw Technol 150:106985
3. Bates S, Hastie T, Tibshirani R (2023) Cross-validation: what does it estimate and how well does it do it? Journal of the American Statistical Association pp 1–12
4. Bennin KE, Keung JW, Monden A (2019) On the relative value of data resampling approaches for software defect prediction. Empir Softw Eng 24:602–636
5. Bennin KE, Tahir A, MacDonell SG, Börstler J (2022) An empirical study on the effectiveness of data resampling approaches for cross-project software defect prediction. IET Software 16(2):185–199
6. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research 16:321–357
7. Cheng Y, Cao G, Wang X, Pan J (2013) Weighted multi-source tradaboost. Chin J Electron 22(3):505–510
8. Chicco D, Tötsch N, Jurman G (2021) The matthews correlation coefficient (mcc) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. BioData mining 14(1):1–22
9. Tachet des Combes R, Zhao H, Wang YX, Gordon GJ (2020) Domain adaptation with conditional distribution matching and generalized label shift. Adv Neural Inf Process Syst 33:19276–19289
10. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research 7:1–30
11. D'Ambros M, Lanza M, Robbes R, (2012) Evaluating defect prediction approaches: a benchmark and an extensive comparison. Empir Softw Eng 17(4):531–577
12. Falkner S, Klein A, Hutter F (2018) Bohb: Robust and efficient hyperparameter optimization at scale. In: International Conference on Machine Learning, PMLR, pp 1437–1446
13. Felix EA, Lee SP (2020) Predicting the number of defects in a new software version. PLoS ONE 15(3):e0229131
14. Fernández A, García S, Galar M, Prati RC, Krawczyk B, Herrera F (2018) Learning from imbalanced data sets, vol 11. Springer
15. Ganchev P, Malehorn D, Bigbee WL, Gopalakrishnan V (2011) Transfer learning of classification rules for biomarker discovery and verification from molecular profiling studies. J Biomed Inform 44:S17–S23
16. Goyal S (2022) Handling class-imbalance with knn (neighbourhood) under-sampling for software defect prediction. Artif Intell Rev 55(3):2023–2064
17. Gui L, Xu R, Lu Q, Du J, Zhou Y (2018) Negative transfer detection in transductive transfer learning. Int J Mach Learn Cybern 9(2):185–197
18. Gupta A, Sharma S, Goyal S, Rashid M (2020) Novel xgboost tuned machine learning model for software bug prediction. In: 2020 International Conference on Intelligent Engineering and Management (ICIEM), IEEE, pp 376–380
19. He H, Bai Y, Garcia EA, Li S (2008) Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence), IEEE, pp 1322–1328
20. Hitz M, Montazeri B (1996) Chidamber and kemerer's metrics suite: a measurement theory perspective. IEEE Trans Software Eng 22(4):267–271
21. Jing X, Wu F, Dong X, Qi F, Xu B (2015) Heterogeneous cross-company defect prediction by unified metric representation and cca-based transfer learning. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pp 496–507
22. Khatri Y, Singh SK (2021) Cross project defect prediction: a comprehensive survey with its swot analysis. Innovations in Systems and Software Engineering pp 1–19
23. Kondo M, German DM, Mizuno O, Choi EH (2020) The impact of context metrics on just-in-time defect prediction. Empir Softw Eng 25(1):890–939
24. Li K, Xiang Z, Chen T, Wang S, Tan KC (2020) Understanding the automated parameter optimization on transfer learning for cross-project defect prediction: an empirical study. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, pp 566–577
25. Li L, Jamieson KG, DeSalvo G, Rostamizadeh A, Talwalkar A (2017) Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In: ICLR (Poster)
26. Li L, Shi K, REN Z (2022) Cross-project defect prediction method based on feature selection and tradaboost. Journal of Computer Applications 42(5):1554
27. Liaw R, Liang E, Nishihara R, Moritz P, Gonzalez JE, Stoica I (2018) Tune: A research platform for distributed model selection and training. arXiv preprint arXiv:1807.05118
28. Liu SM, Chen JH, Liu Z (2023) An empirical study of dynamic selection and random under-sampling for the class imbalance problem. Expert Syst Appl 221:119703
29. Long M, Wang J, Ding G, Sun J, Yu PS (2013) Transfer feature learning with joint distribution adaptation. In: Proceedings of the IEEE international conference on computer vision, pp 2200–2207

30. Malhotra R, Kamal S (2019) An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data. Neurocomputing 343:120–140
31. Meng F, Cheng W, Wang J (2021) Semi-supervised software defect prediction model based on tri-training. KSII Transactions on Internet & Information Systems 15(11)
32. Menzies T, Greenwald J, Frank A (2006) Data mining static code attributes to learn defect predictors. IEEE Trans Software Eng 33(1):2–13
33. Mousaei T (2020) Review on role of quality assurance in waterfall and agile software development. Journal of Software Engineering & Intelligent Systems 5
34. Nam J, Pan SJ, Kim S (2013) Transfer defect learning. In: 2013 35th international conference on software engineering (ICSE), IEEE, pp 382–391
35. Omondiagbe OP, Licorish SA, MacDonell SG (2022) Negative transfer in cross project defect prediction: Effect of domain divergence. In: 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, pp 1–4
36. Pan SJ, Tsang IW, Kwok JT, Yang Q (2011) Domain adaptation via transfer component analysis. IEEE Trans Neural Networks. https://doi.org/10.1109/TNN.2010.2091281
37. Pineau J, Vincent-Lamarre P, Sinha K, Larivière V, Beygelzimer A, d'Alché Buc F, Fox E, Larochelle H (2021) Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). The Journal of Machine Learning Research 22(1):7459–7478
38. Ryu D, Jang JI, Baik J (2015) A hybrid instance selection using nearest-neighbor for cross-project defect prediction. J Comput Sci Technol 30(5):969–980
39. Ryu D, Jang JI, Baik J (2017) A transfer cost-sensitive boosting approach for cross-project defect prediction. Software Qual J 25(1):235–272
40. Sharma U, Sadam R (2023) 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 Syst Softw 195:111522
41. Shrikanth N, Majumder S, Menzies T (2021) Early life cycle software defect prediction. why? how? In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, pp 448–459
42. Tahmoresnezhad J, Hashemi S (2017) Visual domain adaptation via transfer feature learning. Knowl Inf Syst 50(2):585–605
43. Tantithamthavorn C, Hassan AE, Matsumoto K (2018) The impact of class rebalancing techniques on the performance and interpretation of defect prediction models. IEEE Trans Software Eng 46(11):1200–1219
44. Thota MK, Shajin FH, Rajesh P et al (2020) Survey on software defect prediction techniques. International Journal of Applied Science and Engineering 17(4):331–344
45. Wang J, Chen Y, Hao S, Feng W, Shen Z (2017) Balanced distribution adaptation for transfer learning. In: 2017 IEEE international conference on data mining (ICDM), IEEE, pp 1129–1134
46. Wang J, Chen Y, Feng W, Yu H, Huang M, Yang Q (2020) Transfer learning with dynamic distribution adaptation. ACM Transactions on Intelligent Systems and Technology (TIST) 11(1):1–25
47. Wang S, Minku LL, Yao X (2018) A systematic study of online class imbalance learning with concept drift. IEEE transactions on neural networks and learning systems 29(10):4802–4821
48. Witten IH, Frank E, Hall MA, Pal CJ (2016) Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers, San Francisco,. https://doi.org/10.1016/c2009-0-19715-5
49. Wu R, Zhang H, Kim S, Cheung SC (2011) Relink: recovering links between bugs and changes. In: Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, pp 15–25
50. Xu QS, Liang YZ (2001) Monte carlo cross validation. Chemom Intell Lab Syst 56(1):1–11
51. Xu X, Zhang JY, Ma E, Son HH, Koyejo S, Li B (2022) Adversarially robust models may not transfer better: Sufficient conditions for domain transferability from the view of regularization. In: International Conference on Machine Learning, PMLR, pp 24770–24802
52. Xu Z, Pang S, Zhang T, Luo XP, Liu J, Tang YT, Yu X, Xue L (2019) Cross project defect prediction via balanced distribution adaptation based transfer learning. J Comput Sci Technol 34:1039–1062
53. Yan H, Ding Y, Li P, Wang Q, Xu Y, Zuo W (2017) Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2272–2281
54. Yao J, Liu B, Wu Y, Li Z (2023) Multi-source heterogeneous kernel mapping in software defect prediction. Appl Sci 13(9):5526
55. Yu X, Zhou P, Zhang J, Liu J (2017) A data filtering method based on agglomerative clustering. In: SEKE, pp 392–397
56. Yuan Y, Li Y, Zhu Z, Li R, Gu X (2021) Joint domain adaptation based on adversarial dynamic parameter learning. IEEE Transactions on Emerging Topics in Computational Intelligence 5(4):714–723
57. Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q (2020) A comprehensive survey on transfer learning. Proc IEEE 109(1):43–76

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Osayande P. Omondiagbe[1,2] · Sherlock A. Licorish[2] · Stephen G. MacDonell[3]

Sherlock A. Licorish
sherlock.licorish@otago.ac.nz

Stephen G. MacDonell
stephen.macdonell@aut.ac.nz;
stephen.macdonell@otago.ac.nz

[1] Department of Informatics, Landcare Reserach, Lincoln, New Zealand

[2] Department of Information science, University of Otago, Dunedin, New Zealand

[3] Software Engineering Research Lab, Auckland University of Technology, Auckland, New Zealand