Improving Transfer Learning for Cross Project Defect Prediction

Osayande Pascal Omondiagbe 1,1, Sherlock Licorish 2, and Stephen G. MacDonell 2

1Landcare Research
2Affiliation not available

October 31, 2023

Abstract

—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 proposed 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, research to date has failed to consider model performance in light of increasing target data or variances in data sampling. We provide the first investigation of when and to what extent the effect of increasing the target data and using various sampling techniques have 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. 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 five other baseline methods (including W-BDA), 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 adjusting for data imbalance and having an awareness of the effect of the increasing availability of target data in CPDP scenarios.
Improving Transfer Learning for Cross Project Defect Prediction

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

Received: date / Accepted: date

Abstract
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 proposed 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 [31]. 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

Osayande P. Omondiagbe
E-mail: omondiagbep@landcareresearch.co.nz

Sherlock A. Licorish
E-mail: sherlock.licorish@otago.ac.nz

Stephen G. MacDonell
E-mail: stephen.macdonell@aut.ac.nz,stephen.macdonell@otago.ac.nz

1Department of Informatics, Landcare Research, Lincoln, New Zealand
2Department of Information science, Otago University, Dunedin, New Zealand
3Software Engineering Research Lab, Auckland University of Technology, Auckland, New Zealand
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 [22]. 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-project data without relying on cross-project data [46], 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) [62], 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 [53]. Accordingly, CPDP can result in poor model performance, as shown in the literature. For instance, previous work by Zimmermann et al. [62] evaluated CPDP performance on 12 projects with a total of 622 combination experiments. They found that about 622 experiments, only 21 of the pairs resulted in performance better than their baseline. 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 [30]. The two types of distribution difference addressed are marginal and conditional differences [50]. 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 [57].

Previous transfer learning-based CPDP methods focus on addressing the marginal distribution [35, 37], conditional distribution [43] or both [28, 57] (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 [60], 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 [51].

Beyond the treatment of marginal and conditional distributions, the work by Yuhu et al. [5] 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 [13]. Similarly, the work of Ganchev et al. [11] shows a performance increase when the source data is larger compared to the target data. This improvement decreases as target data size increase. 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. [11] experiments are different from CPDP, it has been shown that machine learning robustness and generalisation are fundamentally correlated [56]. 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 [53], thereby leading to negative transfer. Furthermore, the extra target data added to the learning system could result in performance degradation due to data shift, but the original W-BDA only tackles this data shift problem by considering the marginal and conditional distribution difference.

For this reason, we extend the Weighted Balanced Distribution Adaptation (W-BDA) method [50], which works by adjusting the weight of each domain class while adaptively adjusting the importance of both the marginal and conditional distributions. As this method does not take into ac-

---

1 An interference of the previous knowledge with new learning, where one set of events could hurt the overall learning process
count 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 (BAUC), (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 [52] and have been used in defect prediction studies [32, 7, 21, 33, 48]. In fact, these indicators were also used in a related study, which is use here for comparison [57]. In this paper we place greater emphasis on the BAUC because it takes into account the class imbalances and also overcomes bias in binary cases [3].

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 [59] and Hybrid Instance Selection using Nearest Neighbour (HISNN) [41]) and three complex methods (Transfer Component Analysis (TCA) [37], Joint Distribution Adaptation (JDA) [28], Weighted Balanced Distribution Adaptation (W-BDA) [50]). In this work, we addressed class imbalance in our 4 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 [15], Smote [4], Ran-

---

2 https://figshare.com/s/4bf7d85f6787c7c6b2

3 https://figshare.com/s/b1361e93b47f3353170
[38] proposed a software module subset selection method called Peter-Filter, and they used a clustering method. Their method works by combining the source and target software modules, and then a k-means clustering method was used to divide the software modules into groups. The group containing at least one module from the target project was selected. The experiments done by Fayola et al. [38] using 56 static code defect datasets from the PROMISE data repository [1] established that Peter-Filter performed better than the NN-Filter in a CDPD setting. Similar to Peter-Filter, Kazuya et al. [19] proposed a filter method called Density-Based Spatial Clustering (DBSCAN). Their approach works by first combining all the source data, after which the advanced clustering method DBSCAN was used to cluster the software modules into several groups. The groups that do not contain the target data were discarded. Their work shows that their method outperforms the NN-Filter and Peter-Filter on 56 software defect datasets. To further improve the DBSCAN filter, Xiao et al. [59] replaced the clustering algorithm with a more aggressive cluster method called “agglomerative clustering”. Their experimental result shows a small improvement over the DBSCAN method. Zhimin et al. [16] presented an approach that automatically selects suitable training data by using distributional characteristics, such as the mode, mean, median, maximum and minimum to calculate the similarity between the training and test data. None of the methods mentioned above addressed the distribution difference issue, but instead focus on selecting similar source data which may lead to a reduction in the availability of training data.

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. Ying et al. [30] introduce a method called the Transfer Naive Bayes (TNB) method which uses only shared features between the source and target datasets. Experiments performed on different defect datasets from NASA and SOFTLAB showed that TNB performed better than the NN-Filter algorithm. Jaechang at al. [35] proposed an extension of the Transfer Component Analysis (TCA) [37] 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) [28] 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 Zhou et al. [57] 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” [42] and hybrid instance selection using nearest neighbour “HISNN” [41] 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. Duksan et al. [42] indicated that the TCSBoost method could handle cases where there is a high degree of data imbalance. Also, Transfer Learning-based Aging-related bug Prediction (TLP) proposed by Fangyun et al. [40] was used on a cross-project defect prediction project by combining TCA and the random oversampling method. TLP shows a significant improvement over TCA with any re-sampling method.

In understanding the importance of both the marginal and conditional distributions in a transfer learning setting, the work of Mingsheng et al. [28] 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 [28]. The Balanced Distribution Adaptation (BDA) was thus developed by Jindong et al. [50] 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 pre-
diction settings where BDA was seen to outperform other filter and transfer learning-based methods [57]. By treating the classes as balanced across the source and target domain when dealing with an imbalanced dataset, this could result in performance degradation [58]. Based on BDA, Jindong et al. [50] 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.

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 [11], 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 provide justification for 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 modeling performance of an approach that incorporates the source and target ratios in W-BDA compare to other CPDP transfer learning-based methods?

Motivation: The distribution differences of CPDP data are often derived from marginal and conditional distribution differences [57]. 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 the source and target ratios in the W-BDA approach in a transfer learning setting?

Motivation: Though W-BDA was designed to address both distribution differences (marginal and conditional) simultaneously [57], 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 Method

3.1 Cross software defect prediction problem definition

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

3.2 Weighted balanced distribution adaptation

In the CPDP setting, transfer learning aims to adapt both the marginal and conditional distributions between domains [47]. This is done to minimize the distance between \( D_s \) and \( D_t \) as shown in Eq. (1).

\[
D(D_s, D_t) \approx D(P(x_s), P(x_t)) + D(P(y_s|x_s), P(y_t|x_t)) \tag{1}
\]

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 noting that BDA exploits a balance factor \( \mu \) to leverage the different importance of distributions as shown in Eq. (2).

\[
D(D_s, D_t) \approx (1 - \mu)D(P(x_s), P(x_t)) + \mu D(P(y_s|x_s), P(y_t|x_t)) \tag{2}
\]

In Eq. (2) \( \mu \in [0, 1] \) is a balance factor which is used to leverage the distribution differences. When \( \mu \) 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 \( \mu \), 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 \( D_t \) has no labels, it is feasible to evaluate the conditional distribution \( P(y_t|x_t) \). To evaluate the conditional distribution \( P(y_t|x_t) \), the class conditional distribution \( P(x_t|y_t) \) is used to approximate the conditional distribution \( P(y_t|x_t) \).

To calculate the divergences between two marginal and two conditional distributions in Eq. (2), the maximum mean discrepancy (MMD) \([37]\) is used. This is shown in Eq. (3). Since MMD is a non-parametric measurement, MMD has been widely applied to many existing transfer learning approaches \([37]\).

\[
\mathcal{D}(D_s, D_t) \approx (1 - \mu) \left| \frac{1}{n} \sum_{i=1}^{n} x_s - \frac{1}{m} \sum_{j=1}^{m} x_t \right|^2_H \\
+ \mu \sum_{c=1}^{C} \left( \frac{1}{n_c} \sum_{x_s \in D_s^{(c)}} x_s - \frac{1}{m_c} \sum_{x_t \in D_t^{(c)}} x_t \right)^2_H
\]

(3)

In Eq. (3) \( H \) represents 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 \( D_s^{(c)} \) and \( D_t^{(c)} \) denote the software module belonging to class \( c \) in source and target domain, respectively. The \( n_c = |D_s^{(c)}|, m_c = |D_t^{(c)}| \), denoting the number of samples belonging to \( D_s^{(c)} \) and \( D_t^{(c)} \) respectively. The first and second terms in Eq. (3) denote the marginal distribution and conditional distribution distance between domains. By further taking advantage of matrix regularization, Eq. (3) may be formalized as Eq. (4).

\[
\min \text{tr} \left( A^\top X \left( (1 - \mu)M_0 + \mu \sum_{c=1}^{C} M_c \right) X^\top A \right) + \lambda\|A\|_F^2 \\
\text{s.t. } A^\top X H X^\top A = I, \quad 0 \leq \mu \leq 1
\]

(4)

In Eq. (4) \( X \) denotes the input data matrix which is made of \( x_s \) and \( x_t, A \) denotes the transformation matrix, \( I \in \mathbb{R}^{(n+m) \times (n+m)} \) represents the identity matrix, and \( H = I - \frac{1}{n} \mathbf{1} \mathbf{1}^\top \) is the centering matrix. Similar to the work of [37], \( M_0 \) and \( M_c \) are the MMD matrices and can be calculated using Eq. (5) and Eq. (6) below. The first term in Eq. (4) represents a balance factor \( \mu \), while \( \lambda \) is the regularization parameter where \( \|A\|_F^2 \) is the Frobenius norm. Two constraints are involved in Eq. (4): the first constraint ensures that the transformed data (\( A^\top X \)) should preserve the inner properties of the original data, and the second constraint enforces the range of the balance factor \( \mu \).

\[
(M_0)_{ij} = \begin{cases} \frac{1}{n}, & x_i, x_j \in D_s \\ \frac{1}{m}, & x_i, x_j \in D_t \end{cases}
\]

\[
(M_c)_{ij} = \begin{cases} \frac{1}{n_c}, & x_i, x_j \in D_s^{(c)} \\ \frac{1}{m_c}, & x_i, x_j \in D_t^{(c)} \\ 0, & \text{otherwise} \end{cases}
\]

(5)

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

\[
L = \text{tr} \left( A^\top X \left( (1 - \mu)M_0 + \mu \sum_{c=1}^{C} M_c \right) X^\top A \right) \\
+ \lambda\|A\|_F^2 + \text{tr} \left( (I - A^\top X H X^\top A)\phi \right)
\]

Next, the first-order derivative of \( L \) with respect to \( \phi \) are set to 0 and the optimization is transformed into a generalized eigen vector composition problem as shown in the Eq. (8) below:

\[
\left( X \left( (1 - \mu)M_0 + \mu \sum_{c=1}^{C} M_c \right) X^\top + \lambda I \right) A = X H X^\top A \phi
\]

(8)

The optimal transformation matrix \( A \) is obtained by solving Eq. (8) and finding its \( d \) 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 \( P(y|x) \). The proposed approximation by \([50]\) is shown in Eq. (9) below.

\[
\|P(y_s|x_s) - P(y_t|x_t)\|_H^2 = \|P(y_s|x_s) - \frac{n}{n+m} \sum_{c=1}^{C} P(y_{s,c}) P(x_t|y_{s,c})\|_H^2
\]

(9)

In Eq. (9), \( \alpha_s \) and \( \alpha_t \) 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 \( W_c \) for each class is constructed as shown below:

\[
(W_c)_{ij} = \begin{cases} \frac{P(y_i^{(c)})}{n_c}, & x_i, x_j \in D_s^{(c)} \\ \frac{P(y_i^{(c)})}{m_c}, & x_i, x_j \in D_t^{(c)} \\ \sqrt{\frac{P(y_i^{(c)})}{m_c \cdot n_c}}, & x_i \in D_s^{(c)}, x_j \in D_t^{(c)} \\ 0, & \text{otherwise} \end{cases}
\]

(10)

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

\[
\min \text{tr} \left( A^\top X \left( (1 - \mu)M_0 + \mu \sum_{c=1}^{C} W_c \right) X^\top A \right) + \lambda\|A\|_F^2 \\
\text{s.t. } A^\top X H X^\top A = I, \quad 0 \leq \mu \leq 1
\]

(11)

Here, we adjust for W-BDA\(^+\) by setting \( \mu = \mu + \epsilon \), where \( \epsilon = (nt/\text{ns}*100) \), where \( nt \) is 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 Eq. (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 \( c \). Given that both the source and target dataset could have different distribution, and this distribution difference is widened with additional data [59], the new introduced factor(s) helps to further change the weight of each class based on the source/target data when performing distribution adaptation.

We present the W-BDA\(^+\) algorithm in Algorithm 1.

**Algorithm 1 W-BDA\(^+\): Weighted Balanced Distribution Adaptation Plus**

**Input:**
- Source feature matrix: \( X_s \)
- Target feature matrix: \( X_t \)
- Source label vector: \( y_s \)
- dimension \( d \)
- balance factor \( \mu \)
- regularization parameter \( \lambda \)
- number of target data \( n_t \)
- number of source data \( n_S \)

**Output:** Transformation matrix \( A \) and classifier \( f \)

Train a base classifier on \( X_t \) and apply prediction on \( X_t \) to get its soft labels \( y_t \). Construct \( X = [X_s, X_t] \), initialize \( M_0 \) and \( M_s \), using Eq. (10).

repeat

- Solve the eigen vector composition problem in Eq. (11) and use \( d \) smallest eigen vectors to build \( A \).
- Train a classifier \( f \) on \( [A^T X_s, y_s] \)
- Update the soft labels of \( D_t \): \( y_t = f(A^T X_t) \)
- Update matrix \( M_t \), using using Eq. (10) for W-BDA

until Convergence

return Classifier \( f \)

The algorithm time complexity using the Big O notation is \( O(m(n_1 +n_2)) \), where \( n_1 + n_2 \) is the eigenvectors extracted, \( O \) is the order of magnitude and \( m \) is the leading eigenvalues.

4 Experimental Setup

4.1 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 Marco et al. [7], 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 [17]. 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 [45, 23, 12]. 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 [57], where they experimented with the Balanced Distribution Adaptation (BDA) algorithm to solve cross project defect.

- **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 [18]. Each project data consist of 29 static code features.

- **RELINK Dataset**: This dataset [54] 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 [57] and Transfer Component Analysis (TCA) [36].

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 29.7%, comprising 997 software modules without defects and 296 modules with defects. For a detailed description of all the metrics in the AEEEM dataset please refer to [7]. 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 a higher defect density of 50.5% and 39.3% respectively.

4.2 Hyperparameter tuning

The goal of this study is to evaluate the impact of incorporating the source and target ratio 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 [38], we normalize 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. 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 [44], which results in more pessimistic predictions of the test (target in our case) data when compared to typical cross-validation [55].

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 compared 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 [53].

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 [14].

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 success testing 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 [8]. To aid reproducibility, we implemented this algorithm using the Tune library [26]. 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.

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 ≤ 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 ar3 and ar5 projects. 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 ar6 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 [37], Joint Distribution Adaptation (JDA) [28], Bruka Filter [59], Hybrid Instance Selection using Nearest Neighbour (HISNN) [41], Weighted Balanced Distribution Adaptation (W-BDA) [50] 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 Section 2 and 3. 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 [20]. 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

---

### Table 1: Properties of the overall benchmark dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Project</th>
<th>Description</th>
<th># M</th>
<th># DM</th>
<th>DM %</th>
<th>Class</th>
<th>Granularity</th>
<th># F</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEEEM</td>
<td>EQ</td>
<td>OSGi framework</td>
<td>324</td>
<td>129</td>
<td>39.8</td>
<td>Java</td>
<td>Class</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>JDT</td>
<td>IDE development</td>
<td>397</td>
<td>206</td>
<td>20.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LC</td>
<td>Search Engine library</td>
<td>691</td>
<td>64</td>
<td>9.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ML</td>
<td>Task management</td>
<td>1862</td>
<td>245</td>
<td>13.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOFTLAB</td>
<td>ar1</td>
<td>Embedded controller for white-goods</td>
<td>121</td>
<td>9</td>
<td>7.4</td>
<td>C</td>
<td>Function</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>ar2</td>
<td>Washing machine</td>
<td>63</td>
<td>8</td>
<td>12.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ar3</td>
<td>Dish washer</td>
<td>107</td>
<td>20</td>
<td>18.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ar4</td>
<td>Refrigerator</td>
<td>36</td>
<td>8</td>
<td>22.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ar5</td>
<td>Embedded controller for white-goods</td>
<td>102</td>
<td>15</td>
<td>14.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NASA</td>
<td>CM1</td>
<td>Spacecraft instrument</td>
<td>127</td>
<td>12</td>
<td>12.8</td>
<td>C</td>
<td>Function</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>MW1</td>
<td>A zero gravity experiment about combustion</td>
<td>251</td>
<td>25</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PC1</td>
<td>Flight software for earth orbiting satellite</td>
<td>696</td>
<td>55</td>
<td>12.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PC3</td>
<td>Flight software for earth orbiting satellite</td>
<td>1 073</td>
<td>132</td>
<td>12.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RELINK</td>
<td>Apache</td>
<td>Web sever</td>
<td>194</td>
<td>98</td>
<td>50.5</td>
<td>Java</td>
<td>File</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Safe</td>
<td>Security</td>
<td>56</td>
<td>22</td>
<td>39.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zxing</td>
<td>Bar-code scanning library</td>
<td>399</td>
<td>118</td>
<td>29.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

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, P5C 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 [4], adaptive synthetic sampling (ADASYN) algorithm [15], random over-sampling and random under-sampling [10]. 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. 1 below. 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).

4.3 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 (BAUC): BAUC measures model performance, taking into account class imbalances, and it also overcomes bias in binary cases [3]. 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 [46]. G-measure (GM): This is the harmonic mean between Recall and the compliment of False Positive Rate measured [46]. F-measure (F1): This is used for evaluating a binary classification model based on the predictions made for the positive class [92]. We use the BAUC, 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), and to assess the performance of this W-BDA customization against other CPDP transfer learning-based methods (RQ2).

4.4 Statistical testing

For our statistical testing we place greater emphasis on the BAUC because it takes into account the class imbalances and also overcomes bias in binary cases [3]. We compare distributions of BAUC measure of various sampling approaches that may have the same median while their distribution could be very different. We use the non-Parametric Scott-Knott test as recommended by previous studies [6, 34] and used by previous software defect prediction work [46, 12] to identify significant differences (rank) among two or more populations. The test is a top-down bi-clustering approach which is used to group classification models into statistically distinct ranks (with an &alpha;=0.05). It works by first dividing the classification techniques into two ranks based on the mean evaluation metric (i.e., in this paper the BAUC was used to rank the models). The mean BAUC of our 10 repeats for each classification method is divided and ranked. The Scott-Knott test continuously executes and divides the ranks only when the previous divided ranks are statistically significantly different. The test stops when the ranks can no longer be further divided into statistically distinct ranks. This test was selected to overcome the issue of overlapping groups generated by other post hoc tests, e.g., Nemenyi's and Tukey HSD test [8].

5 Results

In this section, we present detailed experimental results for the indicators as proposed in Section 4 (BAUC, AUC, GM and F1). We tuned the base and domain adaptation method in performing comparative analysis towards answering our research questions in turn.

5.1 RQ1. How does the modelling performance of an approach that incorporates the source and target ratios in W-BDA compares to other CPDP transfer learning-based methods?

Table 3 - 6 reports 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 BAUC, 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 W-BDA+ belongs to the top rank group and ranks the first or second on the BAUC indicator. This was more noticeable when no sampling method was used in all experiments.

5.2 RQ2. What is the effect of incorporating the source and target ratios in the W-BDA approach in a transfer learning setting?

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. Table 7 - 10 reports the average values of the four indicators for the two methods on the SOFTLAB, NASA, RELINK and AEEEM dataset. The ”” before each indicator in Table 7 - 10 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 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 BAUC, 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 existing 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 BAUC indicator for all eight experiments.

### 6 Discussion and Implications

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

#### 6.1 RQ1. How does the modelling performance of an approach that incorporates the source and target ratios in W-BDA compare to other CPDP transfer learning-based methods?

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 us-
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 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- Smote, Brukka- Random Undersampling, HISNN- Smote), tend to have a higher prediction accuracy than models which address the distribution differences (TCA, JDA), and methods based on weighting the distribution difference equally such as in the JDA method (Brukka- No Sampling, Brukka- Smote). This pattern 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 the 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 as in the JDA method [28], we observed an 8% to 12% increase in performance for both the original W-BDA and our improved W-BDA+ model when compared to the other model (TCA).

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

<table>
<thead>
<tr>
<th>Projects</th>
<th>Target Dataset</th>
<th>CPDP Classifier - Sampling</th>
<th>Overall Rank</th>
<th>BAUC</th>
<th>AUCCC</th>
<th>GM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>W-BDA</td>
<td>NASA MW1 (253 sample)</td>
<td>JDA- ADASYN</td>
<td>7</td>
<td>0.677</td>
<td>0.721</td>
<td>0.706</td>
<td>0.712</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA- No Sampling</td>
<td>6</td>
<td>0.630</td>
<td>0.687</td>
<td>0.705</td>
<td>0.708</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA- Random Overampling</td>
<td>9</td>
<td>0.699</td>
<td>0.756</td>
<td>0.714</td>
<td>0.726</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA- Random Underampling</td>
<td>8</td>
<td>0.705</td>
<td>0.753</td>
<td>0.715</td>
<td>0.726</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA- Smote</td>
<td>9</td>
<td>0.694</td>
<td>0.759</td>
<td>0.707</td>
<td>0.722</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- ADASYN</td>
<td>12</td>
<td>0.700</td>
<td>0.745</td>
<td>0.714</td>
<td>0.726</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- No Sampling</td>
<td>15</td>
<td>0.698</td>
<td>0.744</td>
<td>0.701</td>
<td>0.722</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- Random Overampling</td>
<td>14</td>
<td>0.696</td>
<td>0.742</td>
<td>0.703</td>
<td>0.721</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- Random Underampling</td>
<td>13</td>
<td>0.687</td>
<td>0.734</td>
<td>0.699</td>
<td>0.713</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- Smote</td>
<td>10</td>
<td>0.676</td>
<td>0.720</td>
<td>0.696</td>
<td>0.709</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- ADASYN</td>
<td>21</td>
<td>0.670</td>
<td>0.717</td>
<td>0.697</td>
<td>0.709</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- No Sampling</td>
<td>4</td>
<td>0.670</td>
<td>0.717</td>
<td>0.699</td>
<td>0.710</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- Random Overampling</td>
<td>5</td>
<td>0.670</td>
<td>0.717</td>
<td>0.699</td>
<td>0.710</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- Random Undersampling</td>
<td>7</td>
<td>0.689</td>
<td>0.736</td>
<td>0.714</td>
<td>0.726</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- Smote</td>
<td>9</td>
<td>0.694</td>
<td>0.759</td>
<td>0.707</td>
<td>0.722</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Projects</th>
<th>Target Dataset</th>
<th>CPDP Classifier - Sampling</th>
<th>Overall Rank</th>
<th>BAUC</th>
<th>AUCCC</th>
<th>GM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>W-BDA</td>
<td>NASA MW1 (253 sample)</td>
<td>JDA- ADASYN</td>
<td>7</td>
<td>0.677</td>
<td>0.721</td>
<td>0.706</td>
<td>0.712</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA- No Sampling</td>
<td>6</td>
<td>0.630</td>
<td>0.687</td>
<td>0.705</td>
<td>0.708</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA- Random Overampling</td>
<td>9</td>
<td>0.699</td>
<td>0.756</td>
<td>0.714</td>
<td>0.726</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA- Random Underampling</td>
<td>8</td>
<td>0.705</td>
<td>0.753</td>
<td>0.715</td>
<td>0.726</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA- Smote</td>
<td>9</td>
<td>0.694</td>
<td>0.759</td>
<td>0.707</td>
<td>0.722</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- ADASYN</td>
<td>12</td>
<td>0.700</td>
<td>0.745</td>
<td>0.714</td>
<td>0.726</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- No Sampling</td>
<td>15</td>
<td>0.698</td>
<td>0.744</td>
<td>0.701</td>
<td>0.722</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- Random Overampling</td>
<td>14</td>
<td>0.696</td>
<td>0.742</td>
<td>0.703</td>
<td>0.721</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- Random Underampling</td>
<td>13</td>
<td>0.687</td>
<td>0.734</td>
<td>0.699</td>
<td>0.713</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- Smote</td>
<td>10</td>
<td>0.676</td>
<td>0.720</td>
<td>0.696</td>
<td>0.709</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- ADASYN</td>
<td>21</td>
<td>0.670</td>
<td>0.717</td>
<td>0.697</td>
<td>0.709</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- No Sampling</td>
<td>4</td>
<td>0.670</td>
<td>0.717</td>
<td>0.699</td>
<td>0.710</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- Random Overampling</td>
<td>5</td>
<td>0.670</td>
<td>0.717</td>
<td>0.699</td>
<td>0.710</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- Random Undersampling</td>
<td>7</td>
<td>0.689</td>
<td>0.736</td>
<td>0.714</td>
<td>0.726</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- Smote</td>
<td>9</td>
<td>0.694</td>
<td>0.759</td>
<td>0.707</td>
<td>0.722</td>
</tr>
</tbody>
</table>

The table shows the performance of different CPDP methods compared to W-BDA+ with various sampling techniques for NASA MW1 dataset. The overall model performance becomes stronger when both distribution differences are addressed adaptively.
setting should be chosen carefully. The Scott-Knott rank of these issues. Also, the sampling techniques used in a CPDP functional distributions to the techniques and not worry about willingness to leave the configuration of the marginal and conditional differences. This is particularly necessary if practitioners are more than relying on addressing only one type of distribution. They should favor the choice of sample dataset to use as source that address data distribution differences. This outcome is also seen in the study of Jindong et al. [50], where our formal statistical testing. We observe that when we incorporate 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 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 favor the choice of sample dataset to use as source rather 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

<table>
<thead>
<tr>
<th>Projects</th>
<th>Target Dataset</th>
<th>CPDP Classifier + Sampling</th>
<th>Overall Rank</th>
<th>BAUC</th>
<th>AUC</th>
<th>GM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELINK</td>
<td>safe (56 sample)</td>
<td>JDA- ADASYN</td>
<td>6</td>
<td>0.540</td>
<td>0.576</td>
<td>0.557</td>
<td>0.444</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA- No Sampling</td>
<td>6</td>
<td>0.539</td>
<td>0.547</td>
<td>0.554</td>
<td>0.457</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA- Random Oversampling</td>
<td>7</td>
<td>0.530</td>
<td>0.542</td>
<td>0.531</td>
<td>0.451</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA- Random Undersampling</td>
<td>6</td>
<td>0.540</td>
<td>0.579</td>
<td>0.574</td>
<td>0.456</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TCA- ADASYN</td>
<td>10</td>
<td>0.440</td>
<td>0.479</td>
<td>0.470</td>
<td>0.368</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TCA- No Sampling</td>
<td>6</td>
<td>0.540</td>
<td>0.559</td>
<td>0.556</td>
<td>0.484</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TCA- Random Oversampling</td>
<td>12</td>
<td>0.450</td>
<td>0.498</td>
<td>0.448</td>
<td>0.324</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TCA- Random Undersampling</td>
<td>7</td>
<td>0.530</td>
<td>0.542</td>
<td>0.510</td>
<td>0.398</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- ADASYN</td>
<td>8</td>
<td>0.516</td>
<td>0.547</td>
<td>0.505</td>
<td>0.409</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- No Sampling</td>
<td>14</td>
<td>0.396</td>
<td>0.429</td>
<td>0.409</td>
<td>0.288</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- Random Oversampling</td>
<td>15</td>
<td>0.306</td>
<td>0.429</td>
<td>0.409</td>
<td>0.288</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- Random Undersampling</td>
<td>11</td>
<td>0.454</td>
<td>0.476</td>
<td>0.496</td>
<td>0.396</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HISNN- Impute</td>
<td>10</td>
<td>0.464</td>
<td>0.510</td>
<td>0.498</td>
<td>0.386</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- ADASYN</td>
<td>4</td>
<td>0.617</td>
<td>0.657</td>
<td>0.632</td>
<td>0.594</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- No Sampling</td>
<td>2</td>
<td>0.628</td>
<td>0.655</td>
<td>0.634</td>
<td>0.567</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- Random Oversampling</td>
<td>2</td>
<td>0.620</td>
<td>0.657</td>
<td>0.638</td>
<td>0.610</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- Random Undersampling</td>
<td>3</td>
<td>0.627</td>
<td>0.653</td>
<td>0.643</td>
<td>0.598</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brukka- Impute</td>
<td>11</td>
<td>0.453</td>
<td>0.498</td>
<td>0.445</td>
<td>0.376</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W-BDA+ - No Sampling</td>
<td>1</td>
<td>0.696</td>
<td>0.719</td>
<td>0.758</td>
<td>0.698</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Projects</th>
<th>Target Dataset</th>
<th>CPDP Classifier + Sampling</th>
<th>Overall Rank</th>
<th>BAUC</th>
<th>AUC</th>
<th>GM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEEEM</td>
<td>EQ (324 sample)</td>
<td>WBDA- No Sampling</td>
<td>2</td>
<td>0.519</td>
<td>0.567</td>
<td>0.507</td>
<td>0.483</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W-BDA+ - No Sampling</td>
<td>1</td>
<td>0.964</td>
<td>0.978</td>
<td>0.983</td>
<td>0.878</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LC (691 sample)</td>
<td>2</td>
<td>0.776</td>
<td>0.805</td>
<td>0.754</td>
<td>0.769</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W-BDA+ - No Sampling</td>
<td>1</td>
<td>0.876</td>
<td>0.842</td>
<td>0.812</td>
<td>0.754</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3DT (997 sample)</td>
<td>1</td>
<td>0.775</td>
<td>0.796</td>
<td>0.743</td>
<td>0.676</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W-BDA+ - No Sampling</td>
<td>1</td>
<td>0.964</td>
<td>0.987</td>
<td>0.978</td>
<td>0.892</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Projects</th>
<th>Target Dataset</th>
<th>CPDP Classifier + Sampling</th>
<th>Overall Rank</th>
<th>BAUC</th>
<th>AUC</th>
<th>GM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOFTLAB</td>
<td>ar3 (63 sample)</td>
<td>WBDA- No Sampling</td>
<td>1</td>
<td>0.676</td>
<td>0.721</td>
<td>0.706</td>
<td>0.712</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W-BDA+ - No Sampling</td>
<td>2</td>
<td>0.636</td>
<td>0.685</td>
<td>0.643</td>
<td>0.589</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ar5 (36 sample)</td>
<td>1</td>
<td>0.676</td>
<td>0.712</td>
<td>0.690</td>
<td>0.620</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W-BDA+ - No Sampling</td>
<td>1</td>
<td>0.677</td>
<td>0.721</td>
<td>0.706</td>
<td>0.712</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Projects</th>
<th>Target Dataset</th>
<th>CPDP Classifier + Sampling</th>
<th>Overall Rank</th>
<th>BAUC</th>
<th>AUC</th>
<th>GM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA</td>
<td>MW1 (251 sample)</td>
<td>WBDA- No Sampling</td>
<td>3</td>
<td>0.692</td>
<td>0.741</td>
<td>0.709</td>
<td>0.669</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W-BDA+ - No Sampling</td>
<td>1</td>
<td>0.736</td>
<td>0.788</td>
<td>0.763</td>
<td>0.698</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CM1 (327 sample)</td>
<td>2</td>
<td>0.702</td>
<td>0.743</td>
<td>0.676</td>
<td>0.654</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W-BDA+ - No Sampling</td>
<td>1</td>
<td>0.746</td>
<td>0.788</td>
<td>0.776</td>
<td>0.719</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Projects</th>
<th>Target Dataset</th>
<th>CPDP Classifier + Sampling</th>
<th>Overall Rank</th>
<th>BAUC</th>
<th>AUC</th>
<th>GM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELINK</td>
<td>safe (56 sample)</td>
<td>WBDA- No Sampling</td>
<td>1</td>
<td>0.698</td>
<td>0.736</td>
<td>0.712</td>
<td>0.699</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W-BDA+ - No Sampling</td>
<td>1</td>
<td>0.719</td>
<td>0.738</td>
<td>0.758</td>
<td>0.688</td>
</tr>
</tbody>
</table>

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

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

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

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

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

formed “HISNN” [41], designed to consider both transfer knowledge and class imbalance simultaneously. Similar results were also seen in the study of Jindong et al. [50], 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 ratios data into the W-BDA method, we reported a higher BAUC score. This high BAUC 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 BAUC when the target dataset was EQ and LC in all sampling techniques for the AEEEM project. Our outcomes here are similar to the work of Zhou et al. [57], who had reported that an earlier version of W-BDA called Balanced Distribution Adapta-

6.2 RQ2. What is the effect of incorporating the source and target ratios in the W-BDA approach in a transfer learning setting?

From our experiment, we observed that when we incorporate the source and target ratios data into the W-BDA method, we reported a higher BAUC score. This high BAUC 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 BAUC when the target dataset was EQ and LC in all sampling techniques for the AEEEM project. Our outcomes here are similar to the work of Zhou et al. [57], who had reported that an earlier version of W-BDA called Balanced Distribution Adapta-

*Ommodiagbe et al.*
data level. We explored this further by using various sampling techniques on the W-BDA method and performed formal statistical testing where Section 5 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 5 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 weighing, and was seen to achieve average improvements in a CPDP setting [57], 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 1 above). The results are shown in Fig 2 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 [53] (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).

7 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 [9]. 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). We were also concerned about threats relating to the datasets and the CPDP techniques studied. To mitigate CPDP techniques threat when comparing to the W-BDA method, we studied four main techniques recently found to be state of the art in transfer-based CPDP [24]. Also, for the sampling techniques used in this study, we used only the techniques commonly used in software engineering, thus, other sampling techniques might return different outcomes. Finally,
while we recognize the threats above, we anticipate that our study here still contributes novel findings to transfer-based CPDP modelling.

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

Acknowledgments

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.

Data and Code Availability

The data and all experimental codes are publicly available and can be retrieved from https://figshare.com/s/b1361e90b47531353170 and, https://figshare.com/s/4fbf7d856f787e7c6b2 respectively. 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.
References


Improving Transfer Learning in CPDP