

Cross-project software defect prediction through multiple learning

Yahaya Zakariyau Bala^{1,3}, Pathiah Abdul Samat¹, Khaironi Yatim Sharif¹, Noridayu Manshor²

¹Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Malaysia

²Department of Computer System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Malaysia

³Department of Computer Science, Faculty of Science, Federal University of Kashere, Gombe, Nigeria

Article Info

Article history:

Received Nov 17, 2022

Revised Sep 13, 2023

Accepted Oct 12, 2023

Keywords:

Attribute selection

Cross-project

Multi-learning

Software defect

Stacking

ABSTRACT

Cross-project defect prediction is a method that predicts defects in one software project by using the historical record of another software project. Due to distribution differences and the weak classifier used to build the prediction model, this method has poor prediction performance. Cross-project defect prediction may perform better if distribution differences are reduced, and an appropriate individual classifier is chosen. However, the prediction performance of individual classifiers may be affected in some way by their weaknesses. As a result, in order to boost the accuracy of cross-project defect prediction predictions, this study proposed a strategy that makes use of multiple classifiers and selects attributes that are similar to one another. The proposed method's efficacy was tested using the Relink and AEEEM datasets in an experiment. The findings of the experiments demonstrated that the proposed method produces superior outcomes. To further validate the method, we employed the Wilcoxon sum rank test at 95% significance level. The approach was found to perform significantly better than the baseline methods.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Pathiah Abdul Samat

Department of Software Engineering and Information System

Faculty of Computer Science and Information Technology, Universiti Putra Malaysia

Jalan Universiti 1, 43400 Serdang, Selangor, Malaysia

Email: pathiah@upm.edu.my

1. INTRODUCTION

One of the most important phases in the creation of high-quality software is the software testing activities. However, software testing activities are highly expensive [1]. Numerous studies found that most software errors are caused by only some few parts of the software modules [2], [3]. To reduce the number of resources required for testing the software project, prior to testing activities, software testing team can use software defect prediction (SDP) tools to predict defect-prone modules in software projects [4]. Various studies have shown that SDP under within-project defect prediction method (WPDP) can effectively identify defect prone modules i.e., where historical defect data used for training and testing is extracted from the same project [4]. WPDP has an obvious challenge when a software project is new or has limited historical defect data [5]. In response to the challenge of WPDP, cross-project defect prediction (CPDP) was introduced i.e., where historical defect data of one project (source) is used to predict defect-prone modules in another project (target) Nam *et al.* [6]. However, although the potential usefulness of the CPDP has been validated by Ma *et al.* [7], Peters *et al.* [8], and Rahman *et al.* [9], CPDP has a poor predictive performance because of the difference

between the source and target projects in terms of programming languages, coding styles and developers' experience. Another factor that also affect the predictive performance of CPDP is the weakness of individual machine learning algorithm used for building prediction model [10]. To address the challenge of poor prediction performance in CPDP method, various studies have proposed difference approaches; [1] proposed transfer cost-sensitive boosting, Liu *et al.* [4] proposed two phase transfer learning, He *et al.* [5] proposed simplified training data method, Sun *et al.* [11] proposed filtering method for selecting suitable source project for CPDP, Chen *et al.* [12] proposed deep learning based CPDP, Zhao *et al.* [13] proposed Manifold feature transformation for CPDP, Lei *et al.* [14] proposed CPDP based on feature selection and distance-weight transfer learning. However, the predictive performance of CPDP is till below applicable level and therefore need improvement.

To improve the predictive performance of CPDP, in this study, we proposed a similar attribute selection and multi-learning method (SASMLM), for cross-project defect prediction. Specifically, to improve the predictive performance of CPDP method by minimizing the difference between source and target projects and also the weakness of the individual classifier to determine the effectiveness of our proposed method, we conducted an experiment using eight open-source datasets from AEEEM and Relink. The results of the experiments showed that the proposed SASMLM achieved better F1_score results compared to the baseline CPDP approaches based on both individual dataset and on average scales. To further confirm that the achievement of our proposed SASMLM is not by chance, we analysed the prediction performance of SASMLM statistically against each baseline method using Wilcoxon rank-sum tests. The statistical result indicated that our proposed SASMLM method outperformed all the baseline method significantly. This study contributes in three ways. This study provides: i) method for reducing the difference between source and target project, ii) method for reducing the impact of weak classifier on the performance of cross-project defect prediction, and iii) details analysis on the impact of using multiple classifiers over single classifier when building model for cross-project defect prediction.

2. RELATED WORK

Briand *et al.* [15] conducted a feasibility study on the CPDP technique, in which an object-oriented project dataset was used to train a machine learning classifier for CPDP. Zimmermann *et al.* [16] motivated by this, experimented extensively with 622 cross-project defect predictions. The outcomes showed that just 3.4% of CPDPs were effective. They concluded that reducing the disparity in distribution between the source and target projects could enhance the CPDP's performance. In addition, the predictive performance of CPDP can be enhanced by selecting appropriate machine learning algorithms. As a result, several studies were conducted to improve CPDP's predictive ability. Selecting the attributes (features) that are the closest to one another is one effective strategy for reducing the difference between two projects. Watanabe *et al.* [17], focusing on attribute level suggested transforming the training set for CPDP using the mean distributional characteristics of attributes from both the source and target projects. Nam *et al.* [6] proposed a trichloroacetic acid (TCA+) that reduces the distribution difference and improves CPDP's predictive performance by transforming the attributes of the source and target projects into a latent space. Zhao *et al.* [13] proposed a manifold-based feature transformation CPDP strategy. The Naïve Bayes (NB) classifier was trained using the transformed source project, and the trained NB was then used for the CPDP. Yuan *et al.* [18] proposed a two-phase method for selecting a suitable CPDP training set. The attributes of the source and target projects were first grouped using the clustering method. To select the appropriate training data for the CPDP, the local density of features, feature class relevance, and feature similarity were then taken into consideration.

In addition, to lessen the gap between the source and target projects, some studies concentrate on instance selection. Turhan *et al.* [19] proposed an approach which uses K-nearest neighbor (KNN) to select most similar instances in source and target projects. He *et al.* [20] selected relevant instances with regard to defect count for CPDP using various similarity techniques. Ma *et al.* [7] suggested a method in which each instance in the source project was given weights based on how similar it was to the target project. A classification model for CPDP was trained using weighted instances. Peters *et al.* [8] proposed the Peters filter, a method for selecting suitable training data for CPDP using K-means clustering technique. Herbold [21] suggested using the EM clustering algorithm and KNN to find similar instances that could be used as CPDP training data. Amasaki *et al.* [22] suggested a strategy in which the appropriate instance for training the CPDP model was obtained through the utilization of density-based spatial clustering of applications (DBSCAN) clustering methods. Li *et al.* [23] suggested a method in which a dictionary was used to learn the instances that were similar between the source and target projects. Ni *et al.* [24] proposed a method for CPDP in which initially, similar instances between the source and the target project were chosen utilizing the cluster technique. Second, the limited label in the target project was used in conjugation with TrAdaboost to select instances from the source project.

3. METHOD

3.1. Framework

CPDP model train on previous records (dataset) of other software project (source) to predict defect on another project (target) to reduce the issue of limited or no previous record. The main challenge is the poor performance resulting from differences between projects and weakness of individual classifier. Selecting most similar attributes between the projects for training and adopting stacking technique instead of individual single classifier can improve the performance of CPDP method.

Based on this, this study proposes a method SASMLM. As shown in Figure 1, the method consists of two main parts. In similar attribute selection part, after taking two different software projects as inputs, one serving as source S and the other as target T, the mean difference (MD) between all the features in the source and target datasets is computed, and features with a minimum MD are extracted from the source and target datasets denoted as training set (TrS) and testing set (TS) for training and testing, respectively. In multi-learning part, in the first stage, a set of base classifiers, KNN, SVM, and RF trained on TrS and tested on TS. In the second stage, the meta-classifier LR is learned from the output of the KNN, SVM, and RF, and the learned meta-classifier (LR) makes a final prediction of TS. Finally, the F1_score is used to evaluate the predictive performance of the proposed method.

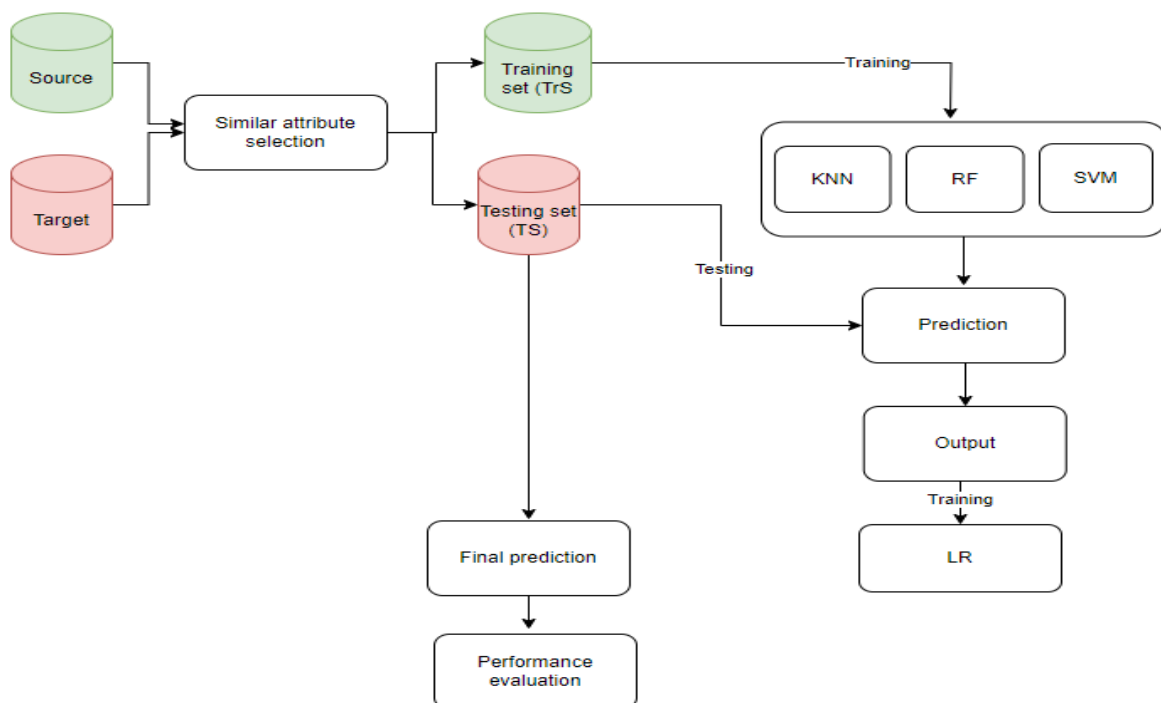


Figure 1. Framework for SASMLM

3.2. Algorithm description

In the initialization phase, (line 1 to 12 in the Algorithm 1), the mean $\hat{f}_j^{(s)}$ of each feature $f^{(s)}$ in the source project S and the mean $\hat{f}_j^{(t)}$ of each feature $f^{(t)}$ in the target project T are computed. To find the similar features between features in the source and target projects, we compute the absolute MD between each feature in the source and the corresponding feature in the target project, for example, mean of feature 1 in source project is subtracted from mean of feature 1 in target project. The absolute values of MD are then arranged in ascending order. We select N top features in MD and used them as common features (CF) between the source and target projects. We then select all features in the source project that are same with the features in the CF and used them as TrS, similarly, all features in the target project that are same with features in CF are selected and used as TS.

In the training phase, (line 14 to 16 in the Algorithm 1), we employed method similar to stacking, KNN, SVM, and RF are trained on TrS and tested on TS. The prediction performance of all the three models, evaluated using F1_score is then used as training set for LR. In the final prediction phase, (line 17 to 19 in the Algorithm 1), the trained LR model is then used to predict defect in the TS. Prediction performance of the LR model was evaluated using F1_score.

Algorithm 1 for proposed SASMLM

Input: S - Source

T - Target

N – The number of selected similar features

Output: F1_score

1: Get all the features in S, mark as $f^{(s)} = \{f_1^{(s)}, f_2^{(s)}, f_3^{(s)} \dots f_m^{(s)}\}$;

2: Get all the features in T, mark as $f^{(t)} = \{f_1^{(t)}, f_2^{(t)}, f_3^{(t)} \dots f_m^{(t)}\}$;

3: **for each** $f^{(s)}$ and $f^{(t)}$

4: Compute the mean, mark as $\hat{f}_j^{(s)}$ and $\hat{f}_j^{(t)}$;

5: **for each** pair of $\hat{f}_j^{(s)}$ and $\hat{f}_j^{(t)}$

6: Compute the absolute difference, marked as $MD = \{f_1^{(s)} - f_1^{(t)}, f_2^{(s)} - f_2^{(t)}, f_3^{(s)} - f_3^{(t)} \dots f_m^{(s)} - f_m^{(t)}\}$;

7: **end for**

8: Rank the values in MD in ascending order;

9: Select N top features in MD, mark as CF;

10: Transpose CF;

11: Select all features from $f^{(s)}$ that correspond with features in transpose CF, mark as Training Set (TrS);

12: Select all features from $f^{(t)}$ that correspond with features in transpose CF, mark as Testing Set (TS);

13: **end for**

14: Train the base models (KNN, SVM and RF) on TrS;

15: Test the trained base models on the TS and calculate their performance in F1_score;

16: Use the trained base models as features to train the second level model (LR);

17: Use the trained LR to perform a final prediction on the TS;

18: Compute the performance of LR using F1_score;

19: Return F1_score;

3.3. Experimental setup

To evaluate the proposed SASMLM systematically, we established the following research questions: i) RQ1: Does the proposed SASMLM performed better cross-project defect prediction than the compared CPDP methods?; ii) RQ3: How does the different number of learners affect the prediction performance of SASMLM?, and iii) RQ4: How does the difference combination of learners affect the prediction performance of our SASMLM?

3.4. Dataset

In this study, we conducted CPDP on eight open-source and commonly used datasets from AEEEM and Relink. Table 1 provides details of these projects. AEEEM dataset was collected by D'Ambros *et al.* [25]. It consists of five projects. Each project in the AEEEM constitutes a total of 71 attributes in different categories, including entropy of change, entropy of source code, source code, previousdefect metrics, and churn of source code. Relink dataset was collected by Wu *et al.* [26]. It consists of three projects, each with 26 attributes.

Table 1. Datasets

Project name	Modules	Features	Defective modules
EQ	325	71	129
JDT	997	71	206
LC	399	71	64
ML	1862	71	245
PDE	1492	71	209
Apache	194	26	98
Safe	56	26	22
ZXing	399	26	118

3.5. Evaluation measures

To evaluate the prediction performance of our approach, we used F-measure which is one of the most used evaluation measures in SDP [27], [28]. F-measure is the harmonic mean of precision and recall. They are defined as follows:

- Precision: determined the ratio of software modules that were accurately predicted to be non-defective to all modules that were predicted to be non-defective.

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

- Recall: determined the ratio between the total number of non-defective modules and the number of modules that were accurately predicted to be non-defective.

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

- F-measure: defined the harmonic representation of precision and recall. The higher F-measure means better performance.

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3)$$

3.6. Baselines

We first experimentally compared SASMLM with four baseline methods in CPDP to determine the effectiveness of our proposed approach. The baseline methods are Watanabe, Burak, TCA+, and MFTCPDP, which were proposed by Nam *et al.* [6], Zhao *et al.* [13], Watanabe *et al.* [17], and Turhan *et al.* [19], respectively. Experiments were conducted on the AEEEM and Relink datasets. The experimental results were measured using F_measurement.

3.7. Evaluation settings

To be in conformity with the previous studies in SDP, we arrange all the datasets in AEEEM and Relink in pairs. For instance, when EQ was used as a source, each of the other projects in AEEEM was used as a target i.e., $EQ \Rightarrow JDT, ML, PDE, \text{ and } LC$. In all the two benchmark datasets, we conducted 26 CPDP experiments from AEEEM and Relink datasets. For the evaluation of the results, we used F-measure (F1_score).

3.8. Statistical significance test

To evaluate whether SASMLM significantly performed better than the compared baseline methods. Like existing SDP studies [7], [24], [29], we employed a nonparametric Wilcoxon signed-rank test at a confidence level of 95%. This test is a non-parametric test use to confirm whether there is a statistical significance difference between two related variables.

4. RESULTS AND DISCUSSION

4.1. Effectiveness of SASMLM

RQ1: Does the proposed SASMLM performed better cross-project defect prediction results than the baseline CPDP methods?

The comparison results of SASMLM against the compared CPDP methods is reported in this section as shown in Tables 2 and 3. From these tables, we can observe that SASMLM achieves better prediction results in most of the combinations against the compared baseline methods based on F1_score. On individual AEEEM dataset, SASMLM won 15, 11, 17, and 18 out of 20 dataset combinations against MFTCPDP, TCA+, Watanabe and Burak respectively. On the Relink datasets, SASMLM won 3, 4, 6, and 5 out of 6 datasets combinations against MFTCPDP, TCA+, Watanabe, and Burak respectively. Also, considering the average results from across 26 prediction combinations, SASMLM performed better than the baseline methods with mean of 0.82 and 0.73 on AEEEM and Relink respectively. To further confirm that the achievement of SASMLM is not by chance, we analysed the prediction performance of SASMLM statistically. Like previous studies [1], [30], [31] we established the following hypothesis:

- H₁₀: SASMLM does not achieve better prediction performance than the four baseline CPDP methods.
- H_{1A}: SASMLM can achieve better prediction performance than the four baseline CPDP methods.

We constructed four groups of Wilcoxon rank-sum tests: SASMLM and MFTCPDP, SASMLM and TCA+, SASMLM and Watanabe, and SASMLM and Burak. At the 95% significance level, the statistical p-values obtained were 0.002 for SASMLM and MFTCPDP, 0.000 for TCA+ and 0.000 for SASMLM and Watanabe, and 0.000 for SASMLM and Burak. The p-values of SASMLM and MFTCPDP, SASMLM and TCA+, SASMLM and Watanabe, and SASMLM and Burak are less than the significant value of 0.05, this indicates that the proposed SASMLM has significantly outperformed all the compared approaches. Therefore, we rejected H₁₀ and accepted H_{1A}. In other words, SASMLM can achieve better F-measure results with statistical significance. A better F-measure indicates that SASMLM can be effective for predicting defects across different projects.

Table 2. Comparison of SASMLM to baseline methods on AEEEM dataset based on F1_score

Source→Target	MFTCPDP	TCA+	Watanabe	Burak	SASMLM
EQ→JDT	0.53	0.46	0.74	0.43	0.87
EQ→LC	0.84	0.48	0.02	0.81	0.95
EQ→ML	0.76	0.39	0.35	0.59	0.94
EQ→PDE	0.72	0.62	0.76	0.53	0.75
JDT→EQ	0.56	0.61	0.69	0.45	0.71
JDT→LC	0.89	0.80	0.83	0.86	0.75
JDT→ML	0.84	0.77	0.78	0.81	0.75
JDT→PDE	0.83	0.80	0.78	0.79	0.88
LC→EQ	0.67	0.55	0.68	0.47	0.65
LC→JDT	0.57	0.73	0.82	0.70	0.73
LC→ML	0.78	0.47	0.81	0.81	0.87
LC→PDE	0.78	0.80	0.81	0.80	0.9
ML→EQ	0.62	0.64	0.68	0.45	0.64
ML→JDT	0.61	0.73	0.83	0.70	0.88
ML→LC	0.88	0.86	0.86	0.86	0.88
ML→PDE	0.82	0.82	0.82	0.79	0.92
PDE→EQ	0.63	0.61	0.69	0.45	0.60
PDE→JDT	0.72	0.74	0.83	0.70	0.85
PDE→LC	0.88	0.79	0.81	0.86	0.92
PDE→ML	0.83	0.79	0.81	0.81	0.91
Mean	0.74	0.67	0.72	0.68	0.82

Table 3. Comparison of SASMLM to baseline methods on relink dataset based on F1_score

Source→Target	MFTCPDP	TCA+	Watanabe	Burak	SASMLM
Safe→Apache	0.71	0.67	0.72	0.57	0.71
ZXing→Apache	0.65	0.67	0.71	0.36	0.63
ZXing→Safe	0.77	0.51	0.72	0.74	0.70
Apache→Safe	0.46	0.57	0.72	0.23	0.78
Apache→ZXing	0.59	0.60	0.65	0.16	0.78
Safe→Zxing	0.65	0.63	0.64	0.60	0.78
Mean	0.64	0.61	0.69	0.44	0.73

4.2. Effect of the number of classifiers (S) on prediction performance of SASMLM

RQ2: How number of classifiers have effect on SASMLM prediction performance?

To investigate the effect of different numbers of classifiers S on the prediction performance of SASMLM, we conducted cross-project defect pre-diction experiments using different numbers of classifiers, starting from S=1 (one classifier), S=2 (two classifiers), and S=3 (three classifiers), as shown in Table 4. The average results across the AEEEM project combinations are also reported in the last row of the corresponding tables. From the table, we can see that the prediction performance of SASMLM is slightly better when S=3 (the number of classifiers used as base learners) than the average performance of other variants.

Table 4. Results of SASMLM based on different number of learners on AEEEM datasets

Source→Target	S=3	S=2	S=1
EQ→JDT	0.71	0.74	0.67
EQ→LC	0.60	0.54	0.48
EQ→ML	0.65	0.59	0.66
EQ→PDE	0.64	0.57	0.52
JDT→EQ	0.73	0.73	0.74
JDT→LC	0.87	0.84	0.88
JDT→ML	0.85	0.80	0.85
JDT→PDE	0.88	0.87	0.88
LC→EQ	0.75	0.75	0.75
LC→JDT	0.88	0.88	0.88
LC→ML	0.9	0.93	0.93
LC→PDE	0.92	0.92	0.92
ML→EQ	0.75	0.75	0.75
ML→JDT	0.88	0.88	0.88
ML→LC	0.95	0.95	0.95
ML→PDE	0.92	0.92	0.92
PDE→EQ	0.75	0.75	0.75
PDE→JDT	0.87	0.88	0.88
PDE→LC	0.94	0.94	0.95
PDE→ML	0.91	0.93	0.93
Mean	0.82	0.81	0.81

4.3. Effect of the difference combinations of classifiers on prediction performance of SASMLM

RQ3: How difference combinations of classifiers have effect on SASMLM prediction performance?

To answer this question, we conducted an experimental investigation on the performance of SASMLM using a combination of KNN and RF, KNN and SVM, and RF and SVM as base learners on AEEEM datasets. The results obtained, as shown in Table 5, show that the SASMLM achieved the highest average F1_score (0.83) using the KNN and RF as base learners compared to the other combinations. Therefore, this result shows that KNN and RF are suitable as base learners for SDP. This agrees with the findings of Akour *et al.* [32] and Goyal *et al.* [33] on RF and KNN, respectively.

Table 5. Results of SASMLM based on combination of different classifiers used as base learners on AEEEM datasets

Source→Target	KNN and RF	KNN and SVM	RF and SVM
EQ→JDT	0.74	0.78	0.71
EQ→ML	0.73	0.49	0.55
EQ→PDE	0.65	0.54	0.51
EQ→LC	0.85	0.39	0.39
JDT→EQ	0.72	0.74	0.74
JDT→ML	0.72	0.84	0.85
JDT→PDE	0.85	0.88	0.87
JDT→LC	0.89	0.83	0.81
ML→EQ	0.75	0.75	0.75
ML→JDT	0.88	0.88	0.88
ML→PDE	0.92	0.92	0.92
ML→LC	0.95	0.94	0.95
PDE→EQ	0.75	0.75	0.75
PDE→JDT	0.88	0.88	0.88
PDE→ML	0.93	0.93	0.92
PDE→LC	0.95	0.94	0.94
LC→EQ	0.75	0.75	0.75
LC→JDT	0.88	0.88	0.88
LC→ML	0.92	0.93	0.93
LC→PDE	0.92	0.92	0.92
Mean	0.83	0.80	0.80

5. CONCLUSION

In this study, we propose a new method for improving the predictive performance of a CPDP called SASMLM. SASMLM first computes the mean vector of all attributes in both the source and target projects, and selects the most similar ones. The selected attributes were then used to train the base learners. The trained base learners were tested on the target project. The results of the base learners are used to train the meta-learner, and the trained meta-learner is used for the final prediction. We conducted experiments to evaluate SASMLM on eight projects from two benchmark datasets, using the F-measure as the evaluation metric. The non-parametric Wilcoxon signed-rank test was used to evaluate the significant difference between SASMLM and the four compared methods. The results show that SASMLM can achieve better prediction performance than competing CPDP methods. Our findings, therefore, show that SASMLM can be useful for SDP, especially when there is a shortage of historical records for a particular project. In future studies, we intend to extend this study by using more classifiers as base learners. Adding new classifiers may help the approach predict more defects.

ACKNOWLEDGMENT




This work was supported by University Putra Malaysia and Tetfund Nigeria (TETF/DASTD/UNIV/GOMBE STATE/TSAS/2019).

REFERENCES




- [1] D. Ryu, J.-I. Jang, and J. Baik, "A transfer cost-sensitive boosting approach for cross-project defect prediction," *Software Quality Journal*, vol. 25, no. 1, pp. 235–272, Mar. 2017, doi: 10.1007/s11219-015-9287-1.
- [2] B. L. Sinaga, S. Ahmad, Z. A. Abas, and A. W. Anggarajati, "The impact of training data selection on the software defect prediction performance and data complexity," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 5, pp. 2903–2921, Oct. 2022, doi: 10.11591/eei.v11i5.3698.
- [3] F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model with rank transformed predictors," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2107–2145, Oct. 2016, doi: 10.1007/s10664-015-9396-2.
- [4] C. Liu, D. Yang, X. Xia, M. Yan, and X. Zhang, "A two-phase transfer learning model for cross-project defect prediction," *Information and Software Technology*, vol. 107, pp. 125–136, Mar. 2019, doi: 10.1016/j.infsof.2018.11.005.

- [5] P. He, Y. He, L. Yu, and B. Li, "An Improved Method for Cross-Project Defect Prediction by Simplifying Training Data," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–18, Jun. 2018, doi: 10.1155/2018/2650415.
- [6] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *2013 35th International Conference on Software Engineering (ICSE)*, IEEE, May 2013, pp. 382–391, doi: 10.1109/ICSE.2013.6606584.
- [7] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," *Information and Software Technology*, vol. 54, no. 3, pp. 248–256, Mar. 2012, doi: 10.1016/j.infsof.2011.09.007.
- [8] F. Peters, T. Menzies, and A. Marcus, "Better cross company defect prediction," in *2013 10th Working Conference on Mining Software Repositories (MSR)*, IEEE, May 2013, pp. 409–418, doi: 10.1109/MSR.2013.6624057.
- [9] F. Rahman, D. Posnett, and P. Devanbu, "Recalling the 'imprecision' of cross-project defect prediction," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, New York, NY, USA: ACM, Nov. 2012, pp. 1–11, doi: 10.1145/2393596.2393669.
- [10] S. Mehta and K. S. Patnaik, "Stacking Based Ensemble Learning for Improved Software Defect Prediction," in *Proceeding of Fifth International Conference on Microelectronics, Computing and Communication Systems Springer*, 2021, pp. 167–178, doi: 10.1007/978-981-16-0275-7_14.
- [11] Z. Sun, J. Li, H. Sun, and L. He, "CFPS: Collaborative filtering based source projects selection for cross-project defect prediction," *Applied Soft Computing*, vol. 99, p. 106940, Feb. 2021, doi: 10.1016/j.asoc.2020.106940.
- [12] D. Chen, X. Chen, H. Li, J. Xie, and Y. Mu, "DeepCPDP: Deep Learning Based Cross-Project Defect Prediction," *IEEE Access*, vol. 7, pp. 184832–184848, 2019, doi: 10.1109/ACCESS.2019.2961129.
- [13] Y. Zhao, Y. Zhu, Q. Yu, and X. Chen, "Cross-Project Defect Prediction Method Based on Manifold Feature Transformation," *Future Internet*, vol. 13, no. 8, p. 216, Aug. 2021, doi: 10.3390/fi13080216.
- [14] T. Lei, J. Xue, Y. Wang, Z. Niu, Z. Shi, and Y. Zhang, "WCM-WTrA: A Cross-Project Defect Prediction Method Based on Feature Selection and Distance-Weight Transfer Learning," *Chinese Journal of Electronics*, vol. 31, no. 2, pp. 354–366, Mar. 2022, doi: 10.1049/cje.2021.00.119.
- [15] L. C. Briand, W. L. Melo, and J. Wust, "Assessing the applicability of fault-proneness models across object-oriented software projects," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp. 706–720, Jul. 2002, doi: 10.1109/TSE.2002.1019484.
- [16] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction," in *Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, New York, NY, USA: ACM, Aug. 2009, pp. 91–100, doi: 10.1145/1595696.1595713.
- [17] S. Watanabe, H. Kaiya, and K. Kaijiri, "Adapting a fault prediction model to allow inter languagereuse," in *Proceedings of the 4th international workshop on Predictor models in software engineering*, New York, NY, USA: ACM, May 2008, pp. 19–24, doi: 10.1145/1370788.1370794.
- [18] Z. Yuan, X. Chen, Z. Cui, and Y. Mu, "ALTRA: Cross-Project Software Defect Prediction via Active Learning and Tradaboost," *IEEE Access*, vol. 8, pp. 30037–30049, 2020, doi: 10.1109/ACCESS.2020.2972644.
- [19] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, no. 5, pp. 540–578, Oct. 2009, doi: 10.1007/s10664-008-9103-7.
- [20] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," *Automated Software Engineering*, vol. 19, no. 2, pp. 167–199, Jun. 2012, doi: 10.1007/s10515-011-0090-3.
- [21] S. Herbold, "Training data selection for cross-project defect prediction," in *Proceedings of the 9th International Conference on Predictive Models in Software Engineering*, New York, NY, USA: ACM, Oct. 2013, pp. 1–10, doi: 10.1145/2499393.2499395.
- [22] S. Amasaki, K. Kawata, and T. Yokogawa, "Improving Cross-Project Defect Prediction Methods with Data Simplification," in *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, IEEE, Aug. 2015, pp. 96–103, doi: 10.1109/SEAA.2015.25.
- [23] Z. Li, X.-Y. Jing, F. Wu, X. Zhu, B. Xu, and S. Ying, "Cost-sensitive transfer kernel canonical correlation analysis for heterogeneous defect prediction," *Automated Software Engineering*, vol. 25, no. 2, pp. 201–245, Jun. 2018, doi: 10.1007/s10515-017-0220-7.
- [24] C. Ni, X. Chen, and W. S. Liu, "Cross-project defect prediction method based on feature transfer and instance transfer," *Journal of Software*, vol. 30, pp. 1308–1329, 2019, doi: 10.13328/j.cnki.jos.005712.
- [25] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: a benchmark and an extensive comparison," *Empirical Software Engineering*, vol. 17, no. 4–5, pp. 531–577, Aug. 2012, doi: 10.1007/s10664-011-9173-9.
- [26] R. Wu, H. Zhang, S. Kim, and S.-C. Cheung, "ReLink," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, New York, NY, USA: ACM, Sep. 2011, pp. 15–25, doi: 10.1145/2025113.2025120.
- [27] W. Wen, B. Zhang, X. Gu, and X. Ju, "An Empirical Study on Combining Source Selection and Transfer Learning for Cross-Project Defect Prediction," in *2019 IEEE 1st International Workshop on Intelligent Bug Fixing (IBF)*, New York, NY, USA: IEEE, Feb. 2019, pp. 29–38, doi: 10.1109/IBF.2019.8665492.
- [28] Y. Z. Bala, P. A. Samat, Y. K. Sharif, and N. Manshor, "Cross-Project Software Defect Prediction," *Journal of Theoretical and Applied Information Technology*, vol. 100, no. 15, pp. 4825–4833, 2022.
- [29] S. Tang, S. Huang, C. Zheng, E. Liu, C. Zong, and Y. Ding, "A novel cross-project software defect prediction algorithm based on transfer learning," *Tsinghua Science and Technology*, vol. 27, no. 1, pp. 41–57, Feb. 2022, doi: 10.26599/TST.2020.9010040.
- [30] Z. Li, J. Niu, X.-Y. Jing, W. Yu, and C. Qi, "Cross-Project Defect Prediction via Landmark Selection-Based Kernelized Discriminant Subspace Alignment," *IEEE Transactions on Reliability*, vol. 70, no. 3, pp. 996–1013, Sep. 2021, doi: 10.1109/TR.2021.3074660.
- [31] Y. Z. Bala, P. A. Samat, K. Y. Sharif, and N. Manshor, "Improving Cross-Project Software Defect Prediction Method Through Transformation and Feature Selection Approach," *IEEE Access*, vol. 11, pp. 2318–2326, 2023, doi: 10.1109/ACCESS.2022.3231456.
- [32] M. Akour, I. Alsmadi, and I. Alazzam, "Software fault proneness prediction: A comparative study between bagging, boosting, and stacking ensemble and base learner methods," *International Journal of Data Analysis Techniques and Strategies*, vol. 9, no. 1, pp. 1–16, 2017, doi: 10.1504/IJDATS.2017.083058.
- [33] R. Goyal, P. Chandra, and Y. Singh, "Suitability of KNN Regression in the Development of Interaction based Software Fault Prediction Models," *IERI Procedia*, vol. 6, pp. 15–21, 2014, doi: 10.1016/j.ieri.2014.03.004.




BIOGRAPHIES OF AUTHORS

Yahaya Zakariyau Bala    received his B.Sc. and M.Sc. degrees in computer science from Adamawa State University Mubi, Nigeria, in 2008 and 2014, respectively, and is currently a Ph.D. student in the Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). He is currently lecturer i with the Department of Computer Science, Faculty of Science, Federal University of Kashere, Nigeria. His research interests include software defect prediction and cross-project software defect prediction. He can be contacted at email: balagombi2@gmail.com or gs61002@student.upm.edu.my.






Pathiah Abdul Samat    received her B.Sc. and M.Sc. degrees in computer science from Universiti Teknologi Malaysia (UTM), in 1996 and 1998, respectively, and Ph.D. degree in Computer Science from Universiti Kebangsaan Malaysia (UKM), in 2012. She is currently a Senior Lecturer with the Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). Her research interests include formal software verification and model checking. She can be contacted at email: pathiah@upm.edu.my.



Khaironi Yatim Sharif    received the Ph.D. degree from the University of Limerick, Ireland. He is currently a Senior Lecturer with the Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. He is also an Adjunct Associate Professor with the Shibaura Institute of Technology, Japan. His research interest includes the area of programmers' information need, particularly identifying programmers' information needs with regards to their task contexts such as software maintenance, program comprehension, code concept mapping, fault localization, and agile development. He can be contacted at email: khaironi@upm.edu.my.



Noridayu Manshor    received her B.Sc., and M.Sc. degrees in computer science from Universiti Putra Malaysia (UTM) and Universiti Teknologi Malaysia (UTM), respectively, and Ph.D. degree in Computer Science from Universiti Saint Malaysia (USM). She is currently a Senior Lecturer with the Department of Computer System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM). Her research interests include pattern recognition, image processing, and computer vision. She can be contacted at email: ayu@upm.edu.my.