Computer Science (CO-SCIENCE)
Volume 6 Issue 1 July 2026
Accreditation **Sinta 3** No. SK : 295/C/C3/KPT/2026

Cross-Project Defect Prediction on AEEEM Using SMOTE–Tomek and Ensemble Learning

Fina Sifaul Nufus[1*], Yan Riyanto[2]

[1, 2]Computer Science, Universitas Nusa Mandiri, Jl Marginda No 545, Depok, Jawa Barat, Indonesia

e-mail: [1]14230025@nusamandiri.ac.id, [2]yan.yrt@nusamandiri.ac.id

(*) Corresponding Author

Article Info: Received: 25-03-2026 | Revised : 26-06-2026 | Accepted : 09-07-2026

Abstracts -*Cross-project defect prediction (CPDP) aims to predict software defects in a target project using data from other projects. However, most existing CPDP approaches rely on complex frameworks, such as transfer learning or sophisticated multi-source integration, making them difficult to reproduce and apply in practical software engineering environments. This study proposes a simple yet integrated hybrid preprocessing pipeline consisting of feature normalization, Principal Component Analysis (PCA), SMOTE–Tomek balancing, and decision threshold tuning to improve CPDP performance on the AEEEM dataset. Experiments were conducted under both single-source and multi-source CPDP scenarios using Random Forest (RF) and Support Vector Machine (SVM) classifiers. Performance was evaluated using the F1 Score and the Area Under the Curve (AUC). The experimental results demonstrate that the proposed approach improves prediction performance, particularly under the multi-source CPDP scenario. Compared with the more complex MSCPDP approach, the proposed method achieved a higher F1-score on four out of five target projects and consistently outperformed MSCPDP on all five projects in terms of AUC. Furthermore, the experimental analysis indicates that decision threshold tuning contributed more significantly to performance improvement than class balancing alone. In contrast, the combination of threshold tuning and SMOTE–Tomek yielded the best overall performance. These findings provide empirical evidence that a simple, reproducible preprocessing pipeline can effectively improve CPDP performance without requiring complex learning frameworks.*
Keywords: CPDP, Imbalanced data, Hybrid preprocessing, SMOTE–Tomek, Threshold tuning

INTRODUCTION

Software has become an essential component in various aspects of modern society, ranging from everyday applications to large-scale industrial systems. As software systems continue to grow in size and complexity, ensuring software quality and reliability becomes increasingly important. One of the major challenges in software engineering is the presence of software defects, which may lead to system failures, increased maintenance costs, and reduced user satisfaction. Therefore, identifying defect-prone software modules at an early stage has become an important research topic to support efficient testing and improve software quality (Kumar et al., 2025), (Nugraha et al., 2021).

Software Defect Prediction (SDP) has been widely investigated as an effective approach for identifying defect-prone modules using historical software metrics and machine learning techniques. However, the performance of SDP is often affected by class imbalance, where defective modules represent only a small proportion of the dataset. Under such conditions, machine learning models tend to be biased toward the majority class, resulting in poor detection of defective modules (Dewi et al., 2023), (Handoko & Aditya, 2025). Furthermore, newly developed software projects frequently lack sufficient historical defect data, making it difficult to construct reliable within-project prediction models.

Cross-Project Defect Prediction (CPDP) addresses this limitation by utilizing defect data from other software projects as training data (SALMAN Saeed et al., 2024). Nevertheless, CPDP introduces additional challenges because the source and target projects often exhibit different data distributions, software characteristics, and defect patterns. Such a distribution mismatch frequently reduces prediction performance if appropriate preprocessing techniques are not employed (Nevendra & Singh, 2022).

Several studies have attempted to improve CPDP performance using various strategies. Hybrid preprocessing methods combining feature selection, dimensionality reduction, and data balancing have demonstrated promising improvements in software defect prediction (Goyal, 2025), (Febrian et al., 2025). More

recently, advanced approaches such as Multi-Source Cross-Project Defect Prediction (MSCPDP) have been proposed to exploit multiple source projects and sophisticated data integration mechanisms for improving prediction performance(Zhao et al., 2022). Likewise, domain adaptation and ensemble-based methods have also achieved competitive results in CPDP (Harianto, 2025). However, most existing approaches rely on relatively complex learning frameworks, making them more difficult to reproduce, implement, and deploy in practical software engineering environments.

In addition to implementation complexity, previous studies still present several important limitations. First, many studies primarily focus on within-project defect prediction or evaluate only a limited number of CPDP scenarios, restricting their generalizability across heterogeneous software projects. Second, although various preprocessing techniques have been investigated individually, relatively few studies integrate normalization, dimensionality reduction, advanced hybrid balancing using SMOTE–Tomek, and decision threshold tuning within a single reproducible CPDP pipeline. Third, existing studies rarely perform ablation analysis to investigate the individual contribution of each preprocessing component, making it difficult to determine which preprocessing strategy contributes most to performance improvement.

Motivated by these limitations, this study provides empirical evidence that a simple yet integrated preprocessing pipeline can achieve competitive and even superior performance compared with more complex CPDP approaches. The proposed pipeline combines feature normalization, Principal Component Analysis (PCA), SMOTE–Tomek balancing, and decision threshold tuning to address both class imbalance and distribution mismatch while maintaining high reproducibility. The experiments are conducted using the widely adopted AEEEM dataset under both single-source and multi-source CPDP scenarios with Random Forest and Support Vector Machine classifiers.

Unlike previous studies that mainly emphasize sophisticated learning architectures, this study demonstrates that carefully designed preprocessing alone can substantially improve CPDP performance. Experimental results show that the proposed approach outperforms the more complex MSCPDP method on four of the five target projects in terms of F1-score and consistently achieves higher AUC across all target projects. Furthermore, an ablation analysis is conducted to examine the individual contribution of each preprocessing component, showing that decision threshold tuning contributes more significantly than balancing alone. These findings provide practical evidence that a simple, reproducible preprocessing pipeline can serve as an effective alternative for software teams seeking robust CPDP models without relying on highly complex frameworks.

RELATED WORK

Previous studies on software defect prediction have focused on improving model performance through data balancing, preprocessing, and machine learning techniques. Handling imbalanced datasets remains a major challenge, as defective modules are typically underrepresented compared to non-defective ones. Various approaches such as SMOTE and hybrid balancing techniques have been proposed to improve the detection of minority classes. In addition, preprocessing methods including normalization and dimensionality reduction are commonly applied to enhance model stability and reduce noise in the data. Widely used machine learning algorithms such as Random Forest and SVM have also demonstrated consistent performance across different datasets. In the context of CPDP, multi-source approaches have been explored to improve prediction performance by leveraging multiple source projects, although these approaches often involve more complex data integration processes. To provide a clearer comparison of existing approaches, Table 1 summarizes several representative studies, including their methods, contributions, and limitations.

Table 1. Literature Review

Author	Method	Contribution	Limitation
(Febrian et al., 2025)	Hybrid feature selection + SMOTE	Achieved AUC up to 0.87	Limited to within-project scenario
(**Ghinaya et al., 2024)**	SVM, RF, XGBoost, ensemble	Improved recall, F1, and AUC	Focused on specific datasets
(**Wei, 2025)**	Hybrid preprocessing	Improved accuracy and reduced noise	Lacks CPDP evaluation
(Goyal, 2025)	Systematic review (hybrid balancing)	Demonstrated effectiveness of hybrid methods	No experimental validation
(Ukpe & Amannah, 2025)	SVM + PCA	High accuracy (94.5%) and F1-score (93.0%)	Limited dataset scope
(Zhao et al., 2022)	Multi-source CPDP (MSCPDP)	Improved F1 and AUC	Complex implementation
(Bala et al., 2024)	CPDP baseline (RF, SVM)	RF achieved F1 ≈ 0.60	No preprocessing applied
(Harianto, 2025)	Domain adaptation + ensemble	Improved CPDP performance	High complexity

(Retnani et al., 2024)	ACO + SMOTE + ensemble	Improved prediction accuracy	Complex pipeline
(Chen et al., 2025)	AutoML-based CPDP	Outperformed multiple baselines	High computational cost

Source: Research Result (2026)

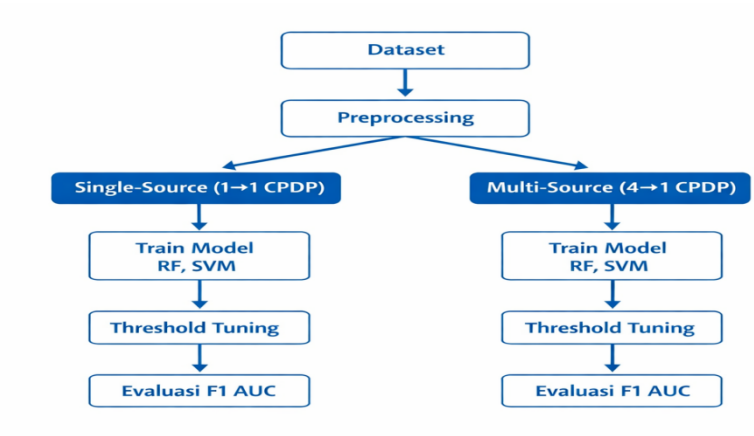
As summarized in Table 1, previous studies have significantly advanced software defect prediction research. Nevertheless, several important limitations remain in CPDP.

Many recent CPDP approaches rely on sophisticated techniques such as domain adaptation, transfer learning, AutoML, or multi-source integration (Zhao et al., 2022), (Harianto, 2025), (Chen et al., 2025). Although these methods achieve competitive performance, they are often difficult to reproduce and computationally expensive. In addition, many hybrid preprocessing studies mainly focus on within-project defect prediction or limited benchmark datasets (Febrian et al., 2025), (Ukpe & Amannah, 2025). Consequently, their effectiveness for heterogeneous CPDP scenarios remains insufficiently validated.

Another limitation is that previous studies generally investigate preprocessing techniques separately. Few studies integrate feature normalization, PCA, SMOTE–Tomek balancing, and decision threshold tuning within a single reproducible CPDP framework. To address these limitations, this study proposes a simple yet integrated preprocessing pipeline combining normalization, PCA, SMOTE–Tomek, and decision threshold tuning. The proposed approach provides empirical evidence that a reproducible preprocessing strategy can achieve competitive CPDP performance without relying on complex learning frameworks.

RESEARCH METHOD

In general, the research methodology consists of four main stages: dataset preparation, data preprocessing, model training, and performance evaluation. The overall workflow of the methodology is illustrated in Figure 1.



Source: Research Result (2026)

Figure 1. Research Flow

Figure 1 presents the CPDP experimental workflow, starting from dataset loading, data preprocessing, and the division into single-source and multi-source scenarios. The process continues with model training using Random Forest (RF) and Support Vector Machine (SVM), followed by threshold tuning, and finally performance evaluation using F1-score and AUC metrics. The diagram also highlights that the target data is not involved in the training or balancing process, thereby preventing the risk of data leakage.

1. Dataset

The dataset used in this study is the AEEEM dataset, which consists of five open-source software projects: EQ, JDT, LC, ML, and PDE. Each dataset contains 61 numerical features representing software metrics and a binary label indicating whether a module is defective or non-defective. This dataset is widely used in software defect prediction research due to its realistic characteristics and variation across projects (Oyo-ita et al., 2024).

Table 2. Dataset Characteristics

Dataset	Modules	Features	Defects	Defect Ratio
EQ	324	61	129	39.8%
JDT	997	61	206	20.7%
LC	691	61	64	9.3%

ML	1862	61	245	13.2%
PDE	1497	61	209	14.0%

Source: Research Result (2026)

The dataset characteristics presented in Table 2 show substantial variations in the number of modules and defect ratios across projects, ranging from 9.3% (LC) to 39.8% (EQ). This wide variation makes the AEEEM dataset well-suited for evaluating preprocessing techniques under realistic class imbalance conditions in CPDP. In particular, projects with very low defect ratios, such as LC, represent the most challenging prediction scenarios because machine learning models tend to be dominated by the majority class. Consequently, improvements achieved on highly imbalanced projects provide stronger evidence of the effectiveness of preprocessing techniques.

2. Experimental Design

The experimental design is conducted under two CPDP scenarios, namely single-source and multi-source CPDP. In the single-source scenario, the model is trained using one source project and tested on a different target project. In contrast, the multi-source scenario trains the model using the combination of the remaining four source projects and evaluates it on the target project. Thus, each target project is evaluated using all available source projects rather than randomly selected combinations. This experimental design enables a comprehensive evaluation of model generalization under different data distribution conditions (Zhao et al., 2022). Each scenario is evaluated under two configurations: baseline and hybrid preprocessing. The baseline configuration applies only standard preprocessing without class balancing or decision threshold tuning, representing conventional CPDP approaches as reported by (Bala et al., 2024). In contrast, the hybrid preprocessing configuration integrates feature normalization, dimensionality reduction using PCA, SMOTE–Tomek balancing, and decision threshold tuning. This experimental setup ensures that any observed performance improvement can be attributed to the proposed preprocessing pipeline rather than differences in classifier selection.

3. Data Preprocessing

Preprocessing plays a crucial role in CPDP due to differences in data distribution across projects. In this study, preprocessing is applied only to source data, while target data is used exclusively for evaluation to avoid data leakage. The preprocessing pipeline consists of normalization, dimensionality reduction, and hybrid balancing. Normalization is applied using standardization techniques to ensure that all features have comparable scales, which is particularly important for algorithms sensitive to feature magnitude (Emma Andini et al., 2022). After normalization, dimensionality reduction is performed using Principal Component Analysis (PCA) to retain 95% of the data variance while reducing noise and feature redundancy. The 95% variance threshold was selected following common practice in software defect prediction studies, as it provides a balance between preserving informative features and reducing feature redundancy. PCA has been shown to improve model performance in high-dimensional datasets by simplifying feature representation (Anju & Judith, 2024). To address class imbalance, a hybrid balancing technique combining SMOTE and Tomek Links is applied. SMOTE generates synthetic samples for the minority class, while Tomek Links remove overlapping samples from the majority class, resulting in a cleaner and more balanced dataset. Compared with standard SMOTE, SMOTE–Tomek is more suitable for CPDP because Tomek Links eliminate ambiguous borderline samples that frequently arise due to distribution mismatch between source and target projects. This combination has been shown to improve classification performance on imbalanced datasets (Astari et al., 2025). To prevent data leakage, all preprocessing steps, including normalization, PCA, and SMOTE–Tomek balancing, are applied exclusively to the source data. The target project is used only during model evaluation.

4. Model Development

This study employs two machine learning algorithms, namely RF and SVM, due to their strong performance in software defect prediction tasks. Random Forest is an ensemble-based algorithm that combines multiple decision trees to improve robustness and reduce overfitting, making it suitable for handling noisy and complex data (Fadia Laila Hidayati et al., 2024). On the other hand, SVM is a supervised learning algorithm that constructs an optimal hyperplane for classification and performs well in high-dimensional feature spaces (Abdulrazak Muhammad Gatawa & Yahaya Isah Shehu, 2025). Both models are selected to provide a fair comparison with previous studies and to evaluate the effectiveness of preprocessing techniques in CPDP scenarios.

5. Threshold Tuning

To improve classification performance on imbalanced datasets, decision threshold tuning is applied instead of using the default threshold value of 0.5. Rather than retraining the classifier, threshold tuning directly optimizes the precision–recall trade-off, making it a computationally efficient strategy for CPDP. Multiple threshold values are evaluated, and the optimal threshold is selected using validation data based on the highest F1-score to reduce

the risk of overfitting. The selected threshold is then applied during evaluation on the target dataset (Abdelhamid & Desai, 2024).

6. Performance Evaluation

The performance of the proposed approach is evaluated using F1-score and Area Under the Curve (AUC). F1-score is used to measure the balance between precision and recall, making it suitable for imbalanced classification problems (Kartika et al., 2025). Meanwhile, AUC evaluates the model's ability to distinguish between defective and non-defective modules across different threshold values, providing a more comprehensive assessment of model performance (Anand et al., 2025). The evaluation is conducted on target datasets under both single-source and multi-source CPDP scenarios, comparing baseline and hybrid preprocessing approaches to analyze the effectiveness of the proposed method.

Overall, the proposed research methodology provides a systematic framework to evaluate the impact of hybrid preprocessing techniques on cross-project defect prediction performance. By integrating normalization, dimensionality reduction, SMOTE–Tomek balancing, and threshold tuning within both single-source and multi-source CPDP scenarios, the experimental design ensures a fair and comprehensive evaluation. The use of widely adopted machine learning models, namely Random Forest and Support Vector Machine, further strengthens the validity of the results. This methodology enables clear analysis of how preprocessing strategies contribute to improving prediction performance, particularly in handling imbalanced data and distribution differences across projects.

RESULTS AND DISCUSSION

This chapter presents the experimental results and performance analysis of the CPDP model applied to the AEEEM dataset. The evaluation is conducted under two main scenarios: single-source CPDP and multi-source CPDP, aiming to assess the effectiveness of the proposed hybrid preprocessing approach in improving defect prediction performance. The hybrid preprocessing includes feature normalization, dimensionality reduction using PCA, class balancing with hybrid SMOTE–Tomek, and decision threshold adjustment. Model performance is evaluated using the F1-score to measure the balance between precision and recall, and AUC to assess the model's discriminative capability.

1. Single-Source CPDP

In the single-source scenario, the model is trained on one source project and tested on a different target project. This represents the simplest form of cross-project defect prediction, where knowledge from one project is transferred to another. However, this approach often faces challenges due to differences in project characteristics, such as metric distributions, number of modules, code complexity, and defect ratios, which can affect model generalization.

To address this, experiments are conducted using two algorithms, Random Forest and Support Vector Machine, under two configurations: baseline and SMOTE-Tomek. The objective is to evaluate the stability of each model when relying on a single source project and to examine whether data balancing techniques effectively improve performance in cross-project settings.

Table 3. Single Source CPDP F1-Score Results

Experiment	Random Forest		SVM	
	Baseline	**Smote-Tomek**	**Baseline**	**Smote-Tomek**
EQ→JDT	0.3802	0.3782	0.4076	0.4072
EQ→LC	0.1920	0.1935	0.2558	0.2348
EQ→ML	0.2426	0.2640	0.2342	0.2361
EQ→PDE	0.2777	0.2754	0.2910	0.2818
JDT→EQ	0.6008	0.6326	0.5984	0.6729
JDT→LC	0.3412	0.3121	0.3086	0.3135
JDT→ML	0.3421	0.3304	0.3640	0.3680
JDT→PDE	0.3862	0.3886	0.4076	0.4000
LC→EQ	0.5724	0.6000	0.6383	0.6410
LC→JDT	0.5679	0.5690	0.4072	0.4227
LC→ML	0.3850	0.2979	0.2955	0.3587
LC→PDE	0.3785	0.2819	0.3043	0.2849
ML→EQ	0.6111	0.6048	0.4930	0.4925
ML→JDT	0.4260	0.4272	0.4105	0.4058
ML→LC	0.3391	0.2477	0.2556	0.2473

ML→PDE	0.3382	0.2835	0.3003	0.2777
PDE→EQ	0.5878	0.6061	0.6575	0.6760
PDE→JDT	0.5048	0.5126	0.4311	0.4386
PDE→LC	0.3478	0.2137	0.2667	0.2551
PDE→ML	0.2647	0.2523	0.2319	0.2342

Source: Research Result (2026)

The F1-score results in the single-source scenario exhibit substantial variability across source–target combinations, indicating that prediction performance is highly dependent on the compatibility between source and target projects. Higher performance is observed for combinations such as JDT→EQ and ML→EQ, whereas EQ→LC and PDE→LC produce considerably lower F1-scores. This variability suggests that single-source CPDP is inherently unstable because differences in feature distributions and defect characteristics often limit the transferability of prediction models across projects.

The impact of SMOTE–Tomek is also inconsistent. While balancing improves prediction performance in several source–target combinations, it decreases performance in others. One possible explanation is that synthetic minority samples generated from the source project may not accurately represent the target project distribution. Consequently, the classifier may learn patterns that are less relevant to the target dataset, leading to reduced generalization performance despite a more balanced training set. These findings indicate that balancing techniques should be applied cautiously in CPDP, particularly when substantial distribution mismatch exists between source and target projects.

Table 4. Single Source CPDP AUC Results.

Experiment	**Random Forest**		**SVM**	
	Baseline	**Smote-Tomek**	**Baseline**	**Smote-Tomek**
EQ→JDT	0.5341	0.5715	0.5917	0.5772
EQ→LC	0.5575	0.5497	0.6755	0.6228
EQ→ML	0.5346	0.5555	0.3941	0.3996
EQ→PDE	0.5411	0.5386	0.5776	0.5614
JDT→EQ	0.6927	0.7148	0.7610	0.7470
JDT→LC	0.7345	0.7020	0.6551	0.6191
JDT→ML	0.6180	0.6160	0.6499	0.6546
JDT→PDE	0.7228	0.7252	0.7315	0.7122
LC→EQ	0.6455	0.6851	0.7493	0.7021
LC→JDT	0.7938	0.8002	0.7172	0.6507
LC→ML	0.6490	0.6593	0.6293	0.7169
LC→PDE	0.7060	0.6028	0.6268	0.5964
ML→EQ	0.6769	0.6735	0.3948	0.3508
ML→JDT	0.6277	0.6064	0.6037	0.5839
ML→LC	0.7413	0.6673	0.6608	0.6336
ML→PDE	0.6874	0.6094	0.6122	0.5836
PDE→EQ	0.6507	0.6292	0.6795	0.7193
PDE→JDT	0.7850	0.7758	0.6905	0.6484
PDE→LC	0.7371	0.6191	0.6590	0.6431
PDE→ML	0.5748	0.5564	0.5632	0.4452

Source: Research Result (2026)

Compared with the F1-score, AUC values are generally more stable across different source–target combinations, indicating that the models maintain reasonable discriminative capability despite variations in classification thresholds. Nevertheless, several project pairs still exhibit performance degradation, suggesting that distribution mismatch affects not only classification decisions but also the underlying separability between defective and non-defective modules. These results further confirm that the effectiveness of CPDP largely depends on the similarity between source and target project characteristics.

Multi-Source CPDP

F1-score is used as the primary metric to evaluate the balance between defect detection and misclassification. The results of the multi-source CPDP scenario are presented in Table 5. The results indicate that multi-source CPDP generally provides more stable performance than the single-source scenario because training data from multiple projects captures a broader range of defect characteristics. Random Forest demonstrates more consistent performance across most target projects, while SVM benefits from SMOTE–Tomek in specific cases, particularly for the EQ target.

However, the effect of SMOTE–Tomek is not uniformly positive. For Random Forest, F1-score decreases on the PDE and LC target projects after balancing. This suggests that synthetic samples generated from multiple source projects may introduce patterns that are not fully representative of the target project's defect characteristics. Consequently, although multi-source training improves overall stability, balancing does not always provide additional benefits and should be applied selectively, particularly for target projects with unique defect distributions.

Table 5. Multi-Source CPDP F1-Score Results

Experiment	Random Forest		SVM	
	Baseline	**Smote-Tomek**	**Baseline**	**Smote-Tomek**
(EQ,JDT,LC,ML)→PDE	0.3908	0.3178	0.3646	0.3501
(EQ,JDT,LC,PDE)→ML	0.3401	0.2861	0.3317	0.3755
(EQ,JDT,ML,PDE)→LC	0.3836	0.2784	0.3071	0.3083
(EQ,LC,ML,PDE)→JDT	0.5196	0.5152	0.4376	0.4525
(JDT,LC,ML,PDE)→EQ	0.5875	0.6168	0.6345	0.6533

Source: Research Result (2026)

The AUC results further confirm that multi-source CPDP provides more stable discriminative performance than single-source CPDP. Although SMOTE–Tomek improves AUC for several target projects, slight performance degradation is still observed in others, indicating that balancing alone cannot completely overcome the distribution mismatch between source and target projects. Overall, combining multiple source projects contributes more consistently to model robustness than applying balancing techniques alone.

Table 6. Multi-Source CPDP AUC Results

Experiment	Random Forest		SVM	
	Baseline	**Smote-Tomek**	**Baseline**	**Smote-Tomek**
(EQ,JDT,LC,ML)→PDE	0.7268	0.6722	0.7194	0.7188
(EQ,JDT,LC,PDE)→ML	0.6807	0.6518	0.6810	0.7302
(EQ,JDT,ML,PDE)→LC	0.7837	0.7178	0.7388	0.7467
(EQ,LC,ML,PDE)→JDT	0.7988	0.7868	0.6796	0.6810
(JDT,LC,ML,PDE)→EQ	0.6503	0.6492	0.6871	0.7034

Source: Research Result (2026)

Overall, the AUC results further confirm that multi-source CPDP provides more stable discriminative performance than single-source CPDP. Although SMOTE–Tomek improves AUC for several target projects, balancing alone cannot completely overcome the distribution mismatch between source and target projects. These findings indicate that combining multiple source projects contributes more consistently to prediction robustness than balancing alone.

2. Comparison with MSCPDP

To evaluate the effectiveness of the proposed approach, a comparison with the MSCPDP method is conducted using the same AEEEM dataset. The benchmark values are taken from previous studies, and the best results from this study are used for comparison. The F1-score comparison is presented in Table 7.

The results show that the proposed model outperforms MSCPDP in most target projects. The comparison results demonstrate that the proposed preprocessing pipeline outperforms the more complex MSCPDP approach on four of the five target projects in terms of F1-score. The largest improvement is observed for EQ, whereas LC remains the only project where MSCPDP achieves better performance. A possible explanation is that LC has the lowest defect ratio (9.3%) among all AEEEM projects, making it an extremely imbalanced target. Under this

condition, the sophisticated source selection and distribution alignment mechanisms employed by MSCPDP may better capture target-specific characteristics than a general preprocessing strategy.

Table 7. Comparison of F1-Score with MSCPDP

Project	MSCPDP	The Best Model in This Study	Difference
EQ	0.3185	0.6533	+0.3348
JDT	0.4218	0.5196	+0.0978
LC	0.4355	0.3836	-0.0519
ML	0.3246	0.3755	+0.0509
PDE	0.3593	0.3908	+0.0315

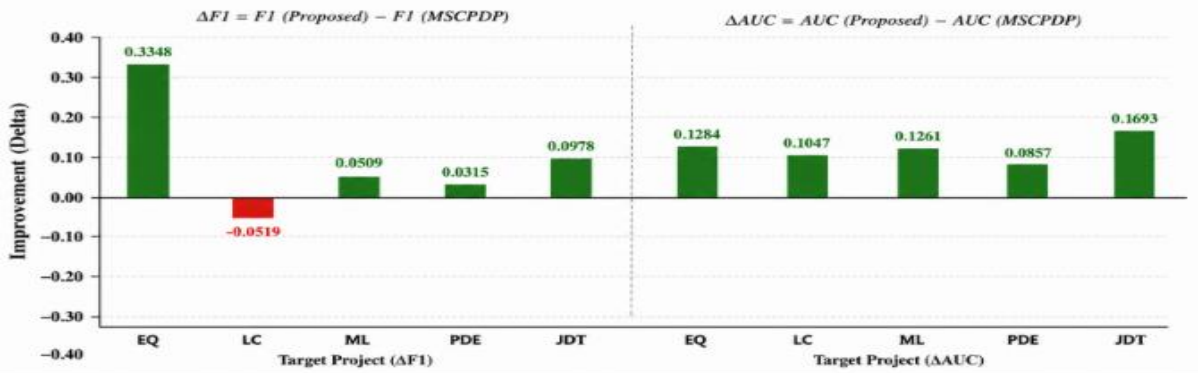
Source: Research Result (2026)

Overall, despite using a simpler approach, the proposed method achieves competitive and often superior performance, demonstrating the effectiveness of multi-source learning combined with appropriate classification models. Furthermore, AUC is used to evaluate the model's overall discriminative capability. The comparison results are shown in Table 8. The results indicate that the proposed model consistently outperforms MSCPDP across all target projects in terms of AUC. The most notable improvement is observed in JDT, where AUC increases from 0.6295 to 0.7988 (+0.1693). Similar improvements are found in ML, LC, EQ, and PDE, indicating better class separation performance.

Table 8. Comparison of AUC with MSCPDP

Project	MSCPDP	The Best Model in This Study	Difference
EQ	0.5750	0.7034	+0.1284
JDT	0.6295	0.7988	+0.1693
LC	0.6790	0.7837	+0.1047
ML	0.6041	0.7302	+0.1261
PDE	0.6411	0.7268	+0.0857

Source: Research Result (2026)



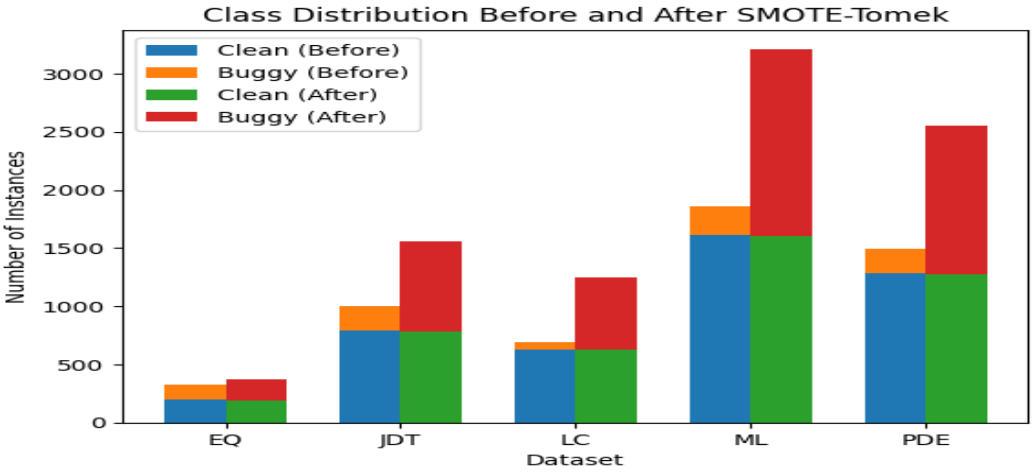
Source: Research Result (2026)

Figure 2. Performance Improvement Compared with MSCPDP

Overall, despite relying on a considerably simpler preprocessing pipeline, the proposed approach achieves higher F1-score on four out of five target projects and consistently outperforms MSCPDP across all target projects in terms of AUC. These findings demonstrate that carefully designed preprocessing can provide competitive CPDP performance without requiring complex transfer learning or sophisticated source selection mechanisms.

3. Analysis of the Effect of Hybrid SMOTE-Tomek

The class distribution before and after applying the hybrid SMOTE–Tomek on the AEEEM dataset is shown in Figure 3.



Source: Research Result (2026)

Figure 3. Class Before and After SMOTE–Tomek

Figure 3 demonstrates that the effectiveness of SMOTE–Tomek varies according to the defect characteristics of the target project. Projects with relatively higher defect ratios, such as EQ (39.8%), generally benefit from class balancing because additional minority-class samples help the classifier better identify defective modules. In contrast, the improvement is less consistent for projects with very low defect ratios, particularly LC (9.3%).

This behavior suggests that, in CPDP, synthetic samples generated from source projects may not always represent the distribution of the target project. As a result, balancing may introduce patterns that are less relevant to highly imbalanced targets, leading to limited or even negative performance gains. Therefore, for projects with extremely low defect ratios, alternative strategies such as source project selection or distribution-aware preprocessing may be more appropriate than relying solely on SMOTE–Tomek.

CONCLUSION

This study demonstrates that integrating feature normalization, PCA, SMOTE–Tomek balancing, and decision threshold tuning effectively improves cross-project defect prediction performance. Overall, the proposed preprocessing pipeline achieves more stable performance under the multi-source CPDP scenario and outperforms the more complex MSCPDP approach on four of five target projects in terms of F1-score and all target projects in terms of AUC. The findings also indicate that decision threshold tuning contributes substantially to performance improvement, while the effectiveness of SMOTE–Tomek depends on the defect characteristics of the target project.

From a practical perspective, multi-source CPDP is recommended over single-source CPDP because it provides more stable prediction performance across heterogeneous projects. Decision threshold tuning should be incorporated as a computationally efficient technique for improving prediction performance without retraining the classifier. However, SMOTE–Tomek should be applied cautiously, particularly for projects with extremely low defect ratios, where source selection strategies may be more effective.

This study has several limitations. The experiments were conducted exclusively on the AEEEM dataset using Random Forest and Support Vector Machine classifiers. Therefore, the generalizability of the proposed approach to other datasets, such as NASA, PROMISE, or industrial software repositories, requires further investigation. Future work may explore adaptive source project selection, lightweight domain adaptation techniques, and automated decision threshold optimization to further improve CPDP performance.

REFERENCES

Abdelhamid, M., & Desai, A. (2024). *Balancing the Scales: A Comprehensive Study on Tackling Class Imbalance in Binary Classification*. http://arxiv.org/abs/2409.19751

Abdulrazak Muhammad Gatawa, & Yahaya Isah Shehu. (2025). Significance of Support Vector Machine Classifier for Predicting Defects in Software Development. *Journal of Science Innovation and Technology Research*. https://doi.org/10.70382/ajsitr.v7i9.010

Anand, K., Jena, A. K., Das, H., Askar, S. S., & Abouhawwash, M. (2025). Software defect prediction using wrapper-based dynamic arithmetic optimization for feature selection. *Connection Science*, *37*(1). https://doi.org/10.1080/09540091.2025.2461080

Anju, A. J., & Judith, J. E. (2024). Hybrid feature selection method for predicting software defect. *Journal of Engineering and Applied Science*, *71*(1), 124. https://doi.org/10.1186/s44147-024-00453-3

Astari, R. A., Sumertajaya, I. M., & Soleh, A. M. (2025). A Hybrid Sampling Approach for Handling Data Imbalance in Ensemble Learning Algorithms. *Scientific Journal of Informatics*, *12*(2), 247–258. https://doi.org/10.15294/sji.v12i2.19163

Bala, Y. Z., Abdul Samat, P., Sharif, K. Y., & Manshor, N. (2024). The influence of machine learning on the

predictive performance of cross-project defect prediction: empirical analysis. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, *22*(4), 830. https://doi.org/10.12928/telkomnika.v22i4.25916

Chen, J., Ding, J., Tan, K. C., Qian, J., & Li, K. (2025). MBL-CPDP: A Multi-Objective Bilevel Method for Cross-Project Defect Prediction. *IEEE Transactions on Software Engineering*, *51*(8), 2305–2328. https://doi.org/10.1109/TSE.2025.3577808

Dewi, M., Saragih, T. H., & Herteno, R. (2023). Penerapan SMOTE-NCL untuk Mengatasi Ketidakseimbangan Kelas pada Klasifikasi Penyakit Jantung Koroner. *Jurnal Informatika Polinema*, *10*(1), 27–34. https://doi.org/10.33795/jip.v10i1.1394

Emma Andini, Faisal, M. R., Rudy Herteno, Nugroho, R. A., Friska Abadi, & Muliadi. (2022). Peningkatan Kinerja Prediksi Cacat Software Dengan Hyperparameter Tuning Pada Algoritma Klasifikasi Deep Forest. *Jurnal Mnemonic*, *5*(2), 119–127. https://doi.org/10.36040/mnemonic.v5i2.4793

Fadia Laila Hidayati, Z., Abadi, F., Reza Faisal, M., & Kartini, D. (2024). Improving Performance of Random Forest Algorithm Using Abc Feature Selection for Software Defect Prediction. *Jurnal CoreIT*, *10*(1), 32–39. https://doi.org/10.24014/coreit.v10i1.29283

Febrian, M. M., Saputro, S. W., Saragih, T. H., Abadi, F., & Herteno, R. (2025). Hybrid Feature Selection and Balancing Data Approach for Improved Software Defect Prediction. *Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics*, *7*(2), 232–244. https://doi.org/10.35882/ijeeemi.v7i2.67

Ghinaya, H., Herteno, R., Faisal, M. R., Farmadi, A., & Indriani, F. (2024). Analysis of Important Features in Software Defect Prediction Using Synthetic Minority Oversampling Techniques (SMOTE), Recursive Feature Elimination (RFE) and Random Forest. *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, *6*(3), 276–288. https://doi.org/10.35882/jeeemi.v6i3.453

Goyal, S. R. (2025). A systematic review on AI based class imbalance handling in software defect prediction. *Results in Engineering*, *27*, 106578. https://doi.org/10.1016/j.rineng.2025.106578

Handoko, C. B., & Aditya, C. S. K. (2025). Penerapan Teknik SMOTE Dalam Mengatasi Imbalance Data Penyakit Diabetes Menggunakan Algoritma ANN. *Smart Comp: Jurnalnya Orang Pintar Komputer*, *14*(1), 13–20. https://doi.org/10.30591/smartcomp.v14i1.7045

Harianto, S. (2025). The Improving Cross-Project Software Defect Prediction with CORAL-Based Domain Adaptation and Ensemble Learning. *Telematika*, *22*(1). https://doi.org/10.31315/telematika.v22i1.14939

Kartika, N. P., Herteno, R., Budiman, I., & Nugrahadi, D. T. (2025). *Comparative Performance Evaluation of Linear , Bagging , and Boosting Models Using BorutaSHAP for Software Defect Prediction on NASA MDP Datasets*. *6*(6), 5821–5836.

Kumar, N., Sangwan, O. P., & Beniwal, S. (2025). *Software defect prediction using machine learning techniques*. 040004. https://doi.org/10.1063/5.0299059

Nevendra, M., & Singh, P. (2022). Cross-Project Defect Prediction with Metrics Selection and Balancing Approach. *Applied Computer Systems*, *27*(2), 137–148. https://doi.org/10.2478/acss-2022-0015

Nugraha, A. S., Faisal, M. R., Abadi, F., & ... (2021). Deep Neural Network on Software Defect Prediction. *... and Software ...*, *02*(02), 82–89. https://jurnalmahasiswamipa.ulm.ac.id/index.php/integer/article/view/44%0Ahttps://jurnalmahasiswamipa.ulm.ac.id/index.php/integer/article/download/44/20

Oyo-ita, E. U., Edim, E. A., Otiko, A., & Izuki, D. E. (2024). *Improving model performance for software defect detection and prediction using ensemble method and cross validation techniques Improving model performance for software defect detection and prediction using ensemble method and cross validation techniques*. *August*.

Retnani, W. E. Y., Furqon, M. 'Ariful, & Setiawan, J. (2024). Improving Software Defect Prediction With a Combination of Feature Selection Based On Ant Colony Optimization and Ensemble Technique. *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*. https://doi.org/10.22219/kinetik.v9i4.2038

SALMAN Saeed, M., Salman Saeed, M., Saleem, M., & Saeed, S. (2024). Cross Project Software Defect Prediction Using Machine Learning: A Review. *Pp*, *3*(June), 52. https://www.researchgate.net/publication/374557048

Ukpe, K. C., & Amannah, C. I. (2025). Hybrid Software Model for Defect Detection and Cost Evaluation Using Support Vector Machine Algorithm. *Journal of Computer and Communications*, *13*(04), 244–264. https://doi.org/10.4236/jcc.2025.134016

Wei, X. (2025). Research on Preprocessing Techniques for Software Defect Prediction Dataset Based on Hybrid Category Balance and Synthetic Sampling Algorithm. *Procedia Computer Science*, *262*, 840–848. https://doi.org/10.1016/j.procs.2025.05.117

Zhao, Y., Zhu, Y., Yu, Q., & Chen, X. (2022). *SS symmetry Cross-Project Defect Prediction Considering Multiple Data*. 1–18.