A LIGHTWEIGHT AND PRACTICAL PIPELINE FOR CROSS-PROJECT DEFECT PREDICTION USING METRIC-BASED LEARNING

Novia Heriyani[1*]; Agus Subekti[2]

Computer Science[1, 2]
Universitas Nusa Mandiri, Jakarta,Indonesia[1, 2]
https://www.nusamandiri.ac.id/[1, 2]
14230034@nusamandiri.ac.id[1*], agus@nusamandiri.ac.id[2]
(*) Corresponding Author

Abstract— *Cross-Project Defect Prediction (CPDP) addresses the scarcity of defect data in new software projects by transferring knowledge from existing ones. However, domain shift between projects remains a major challenge. This study introduces a lightweight and practical CPDP pipeline based on traditional metric features, integrating domain adaptation (CORAL, TCA, TCA+), feature selection, and resampling techniques. Through 120 configurations evaluated on multiple PROMISE datasets, we found that combining TCA or TCA+ with Synthetic Minority Over-sampling Technique combined with Edited Nearest Neighbors (SMOTEENN) consistently improved F1-Score and Recall on imbalanced datasets. LightGBM demonstrated the most stable performance across projects, while Logistic Regression yielded the highest MCC in specific cases. Principal Component Analysis (PCA) visualizations supported the effectiveness of domain alignment. The proposed pipeline offers a reproducible, cost-efficient alternative to deep learning models and provides actionable insights for defect prediction in resource-constrained environments.*

Keywords: *Cross-Project Prediction, Domain Adaption, Machine Learning, Metric-Based Feature, Smoteenn.*

Intisari— *Cross-Project Defect Prediction* (CPDP) menjadi solusi untuk mengatasi keterbatasan data cacat pada proyek perangkat lunak baru dengan mentransfer pengetahuan dari proyek lain. Namun, pergeseran domain antar proyek masih menjadi tantangan utama. Studi ini memperkenalkan pipeline CPDP yang ringan dan praktis berbasis fitur metrik tradisional, yang mengintegrasikan adaptasi domain (CORAL, TCA, TCA+), seleksi fitur, dan teknik resampling. Dari 120 konfigurasi yang dievaluasi pada beberapa dataset PROMISE, kombinasi TCA atau TCA+ dengan SMOTEENN terbukti secara konsisten meningkatkan F1-Score dan Recall pada dataset yang tidak seimbang. Model LightGBM menunjukkan performa paling stabil lintas proyek, sementara Regresi Logistik mencatat nilai MCC tertinggi pada beberapa kasus. Visualisasi PCA mengonfirmasi efektivitas penyelarasan domain. Pipeline yang diusulkan menawarkan alternatif yang dapat direproduksi dan hemat biaya dibanding pendekatan berbasis deep learning, serta memberikan wawasan praktis bagi implementasi prediksi cacat di lingkungan dengan sumber daya terbatas.

Kata Kunci: *Prediksi Lintas Proyek, Adaptasi Domain, Pembelajaran Mesin, Fitur Berbasis Metrik, Smoteenn.*

INTRODUCTION

Software Defect Prediction (SDP) plays an important role in improving software quality by identifying modules at risk of defects before the testing phase (P. S. Kumar, Nayak, & Behera, 2022). Effective defect prediction enables testers to prioritize testing efforts, reduce costs, and enhance defect detection efficiency (Stradowski & Madeyski, 2024). The traditional approach, Within-Project Defect Prediction (WPDP), relies on historical data from the same project. However, this approach is not feasible for new projects without defect history (Tao et al., 2024). In such cases, Cross-Project Defect Prediction (CPDP) becomes more practical, as it transfers knowledge from existing projects to a new target project (Sotto-Mayor & Kalech, 2024).

Early CPDP studies, such as the influential work of Zimmermann et al. (2009), highlighted the difficulty of transferring models across projects due to substantial differences in feature distributions (Zimmermann, Nagappan, Gall, Giger, & Murphy, 2009). Subsequent research further emphasized this challenge, commonly referred to as domain shift, which occurs when the statistical properties of source and target projects differ significantly (Song et al., 2024; Vescan, Găceanu, & Şerban, 2024). Domain shift remains one of the central barriers to achieving reliable CPDP performance.

To overcome this problem, recent studies explore semantic features through Abstract Syntax Tree (AST) representation combined with deep learning models such as Long Short-Term Memory (LSTM) and Convolutional Neural Network-Bidirectional LSTM (CNN-BiLSTM) (Tao et al. 2024). reported a 2.9% improvement in F-measure through the integration of AST and LSTM, while the CNN-BiLSTM approach resulted in a 25% improvement in F1-score (Farid et al. 2021). Although effective, such methods rely on specialized toolchains and high resources, as well as a complex parameter tuning process (Ghinaya, Herteno, Faisal, Farmadi, & Indriani, 2024; Tao et al., 2024).

In light of these limitations, this study proposes a lightweight CPDP pipeline based on traditional metric features. Unlike resource-intensive deep learning models, the proposed approach is computationally efficient and easier to implement. The pipeline integrates domain adaptation techniques such as CORrelation ALignment (CORAL), Transfer Component Analysis (TCA), and its variant TCA+, along with feature selection methods (SelectKBest, Recursive Feature Elimination) and resampling strategies (SMOTEENN). It further leverages classical machine learning models, including Logistic Regression, Random Forest, and LightGBM, which remain competitive in many predictive tasks.

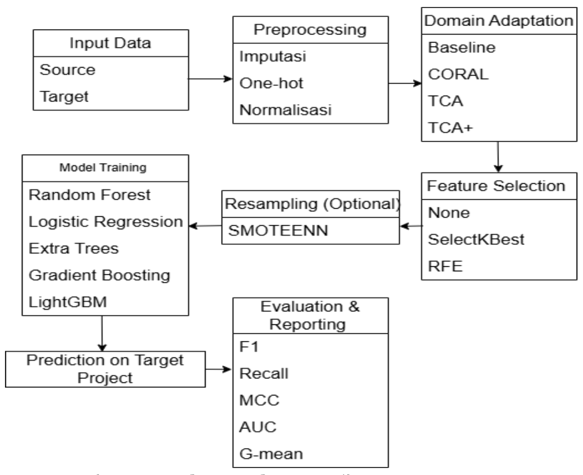
Previous studies have shown the effectiveness of these components individually. For instance, TCA has been demonstrated to harmonize feature distributions and significantly improve AUC (Farid, Fathy, Eldin, & Abd-Elmegid, 2021; Haldar & Fernando Capretz, 2024; Ren, Peng, Zheng, Zou, & Gao, 2022). Similarly, the combination of Synthetic Minority Oversampling Technique (SMOTE) with feature selection has been effective in addressing class imbalance, improving G-mean and recall (Ghinaya et al., 2024; Sharma & Sadam, 2022; Tong, Liu, Wang, & Li, 2019).

Building upon these insights, this study systematically evaluates 120 configurations of domain adaptation, feature selection, resampling, and classifiers across five open-source PROMISE datasets. The contribution of this work lies not in introducing new algorithms, but in the systematic construction and large-scale evaluation of a modular CPDP pipeline that unifies established techniques in a reproducible framework. This structured approach provides empirical insights into which configurations perform best under various conditions, offering practical guidance and a cost-efficient alternative to deep learning models for defect prediction in resource-constrained environments.

MATERIALS AND METHODS

To systematically evaluate the effectiveness of metric-based learning approaches for CPDP, this study constructs an experimental pipeline that combines various domain adaptation techniques, feature selection strategies, resampling methods, and classical machine learning models. The objective of this pipeline is to develop a lightweight yet practical framework that can be implemented in real-world environments with limited computational resources. Each component in the pipeline was chosen based on its prior success in handling specific challenges in CPDP, such as domain shift and class imbalance. The overall methodological framework is designed to allow a comprehensive analysis of how different configurations affect prediction performance across heterogeneous software projects.

The process starts from the loading of source and target project data, followed by pre-processing stages such as imputation and normalization. The source data is then processed through domain adaptation techniques such as CORAL, TCA, and TCA+ to harmonize the distribution between domains. After that, feature selection was performed to filter out the most relevant attributes, followed by an optional resampling technique SMOTEENN to handle class imbalance. The transformed data was used to train five classification models. The trained models were then used to predict defects in the target project. All evaluation results were extracted using five key metrics: F1-score, Recall, Matthews Correlation Coefficient (MCC), and Area Under the Receiver Operating Characteristic Curve (AUC).



Source: (Research result, 2025)
Figure 1. CPDP Flow

1. Data Pre-Processing

The data used comes from open-source projects in the PROMISE repository, including Camel, Log4j, Xalan, Synapse, and Xerces. Each dataset contains software metric features as well as labels for the number of defects (bugs) per module. The labels were converted into a binary format: 1 if there is at least one defect, and 0 if there is none. Non-informative columns such as name, version, and name.1 were removed.

Missing numerical features were imputed using the mean value, while categorical features (if present) were imputed using the mode and encoded with one-hot encoding. All features were then normalized using StandardScaler to avoid dominance of certain features and ensure stability of model training.

Table 1. Dataset Statistic

Dataset	Total Instances	Buggy	Non-Buggy	% Buggy
Camel-1.6	956	188	777	19.48
Log4j-1.2	205	189	16	92.20
Xerces-1.4	588	437	151	74.32
Xalan-2.7	909	898	11	98.79
Synapse-1.2	256	86	170	33.59

Source: (Research result, 2025)

Table 1 presents the summary statistics of the five project datasets used in this experiment. It can be seen that most of the datasets have an unbalanced label distribution, with the ratio of buggy modules ranging from 19% to almost 99%. This confirms the need for the application of class balancing techniques such as SMOTEENN in this CPDP experiment pipeline.

2. Domain Adaptation

Given the inherent heterogeneity between source and target projects in CPDP, discrepancies in feature distributions commonly referred to as domain shift pose a significant challenge to model generalization. To address this issue, the present study employs three domain adaptation techniques: CORAL, TCA, and its enhanced variant, TCA+. CORAL operates by aligning the covariance structures of the source and target domains through a linear transformation, independent of target labels.

TCA mitigates distributional differences by projecting the data into a common latent space using a kernel-based optimization framework. TCA+ further refines this process by introducing a preliminary PCA step to suppress noise and enhance the salience of informative features. Notably, all domain adaptation procedures are applied exclusively to the feature space, without utilizing any label information from the target domain.

3. Fiture Selection

After domain adaptation, feature selection is performed to improve model efficiency and reduce the risk of overfitting. Two methods are used: SelectKBest based on chi-square score, and RFE based on Logistic Regression. SelectKBest selects the top 20 features based on their statistical relationship with the label, while RFE iteratively removes features with the lowest contribution to model performance. Feature selection was performed only on the source data, and then the same transformation was applied to the target data.

4. Resampling

A common problem caused by class imbalance with the amount of non-buggy data far outnumbering buggy in CPDP, can be resampled using the SMOTEENN method. This technique combines SMOTE, which generates synthetic data for the minority class, and Edited Nearest Neighbors (ENN), which cleans up potentially noisy majority data. Resampling is only applied to the source data after domain adaptation and feature selection.

5. Model Training and Testing

The processed source data was trained using five classical machine learning algorithms: Random Forest, Logistic Regression, Gradient Boosting, Extra Trees, and LightGBM. Models were implemented using default settings with minor parameter adjustments (n_estimators=100, max_iter=1000, random_state=123, n_jobs=-1) for consistency and efficiency. To evaluate the impact of various configurations on defect prediction performance, a total of 120 combinations were constructed from different domain adaptation, feature selection, and resampling techniques. All experiments were conducted using Python 3.10 in a Jupyter Notebook environment with key libraries such as scikit-learn, imbalanced-learn, and LightGBM. In each test case, one dataset served as the target (testing) project, while others were used for training, reflecting a realistic CPDP scenario with no labeled data in the target. A concise summary of the experimental environment, tools, datasets, and parameter configurations used throughout the study is provided in Table 2.

Table 2. Summary of Experimental Environment and Configurations

Aspect	Details
Programming Language	Python 3.10
Development Environment	Jupyter Notebook
Libraries Used	scikit-learn, imbalanced-learn, lightgbm

Aspect	Details
Datasets	PROMISE repository: Camel, Log4j, Xalan, Synapse (train), Xerces (target)
CPDP Setting	One project as target; others as source; no label used from target
Models Used	Random Forest, Logistic Regression, Gradient Boosting, Extra Trees, LightGBM
Adjusted Parameters	n_estimators=100, max_iter=1000, random_state=123, n_jobs=-1
Hyperparameter Tuning	Not performed (for reproducibility and efficiency)
Total Configurations	120 (4 domain adaptation × 3 selection × 2 resampling × 5 classifiers)

Source: (Research result, 2025)

Although no hyperparameter tuning was conducted to ensure replicability and computational efficiency, we acknowledge that tuning parameters such as tree depth, learning rate, or regularization terms could potentially enhance model performance. Techniques like grid search or Bayesian optimization might improve predictive accuracy, especially for models like LightGBM or Random Forest. However, this study focuses on demonstrating the robustness of the proposed pipeline under practical constraints, leaving tuning as a direction for future work.

6. Performance Evaluation

Model evaluation is performed by comparing the predicted results on the target project against the actual label. Five evaluation metrics were used: F1-score, Recall, MCC, and AUC. F1-score measures the balance between precision and recall, providing a harmonious representation of both metrics(P. H. Kumar & Bhat, 2024). F1-score provides a value between 0 and 1, where higher values indicate better performance. This metric is particularly useful in the context of software defect prediction as it provides a balance between the model's ability to identify defects (recall) and the accuracy of positive predictions (precision).(Albattah & Alzahrani, 2024).

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (1)$$

Recall indicates the model's ability to detect defect classes by measuring the proportion of true positives identified out of all actual positive instances. Recall is very important in software defect prediction as it measures the model's ability to capture all modules that are actually defective.

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

MCC provides an overall performance representation, especially on unbalanced data, by considering all elements of the confusion matrix. MCC yields a value between -1 and +1, where +1 indicates a perfect prediction, 0 indicates a random prediction, and -1 indicates a completely wrong prediction. MCC is more informative and reliable than accuracy and F1-score, especially on unbalanced datasets (Chicco & Jurman, 2020).

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP(TN+FN)}} \quad (3)$$

AUC measures the model's ability to discriminate classes probabilistically by evaluating the area under the ROC (Receiver Operating Characteristic) curve. AUC provides a value between 0 and 1, where a value of 0.5 indicates random performance, and a value close to 1 indicates excellent discrimination ability (Yang et al., 2021).

$$AUC = \int_0^1 TPR\left(FPR^{-1}(x)\right)dx \quad (4)$$

To enable fair comparison across configurations and projects, we adopt a composite score that aggregates multiple evaluation metrics into a single value. Let $M = \{m_1, m_2, \dots, m_k\}$ be the set of evaluation metrics considered (e.g., F1-score, Recall, AUC, and MCC). For each configuration ccc, the composite score is computed as:

$$CS_{(c)} = \frac{1}{k}\sum_{i=1}^{k} \widehat{m}_i(c) \quad (5)$$

Where $\widehat{m}_i(c)$ is the normalized value of metric m_i for configuration c. Each metric is min–max normalized across all configurations within the same dataset to ensure comparability:

$$\widehat{m}_{i(c)} = \frac{m_{i(c)} - min(m_i)}{max(m_i) - min(m_i)} \quad (6)$$

This normalization maps all metrics to the range [0,1]. The composite score thus represents the average relative performance of a configuration across all metrics.

RESULTS AND DISCUSSION

This experiment was designed to evaluate the effectiveness of traditional metric-based CPDP pipelines. A total of 120 combinations were explored of variations of four domain adaptation methods Baseline, CORAL, TCA, TCA+, two resampling techniques no resampling and

SMOTEENN, three feature selection approaches no selection, SelectKBest, and RFE, as well as five machine learning algorithms Random Forest, Logistic Regression, Gradient Boosting, Extra Trees, and LightGBM. The evaluation was conducted on four target datasets: Camel, Log4j, Synapse, and Xalan using five performance metrics: F1-Score, Recall, MCC, and AUC.

1. Best Configuration for each Project

To provide an overview of the best-performing configurations, Table 3 summarizes the highest composite score achieved for each target project. Composite Score is calculated from the average of four key metrics: F1-score, Recall, AUC, and MCC. These configurations represent the optimal combination of domain adaptation, feature selection, classifier, and resampling strategy in the context of CPDP. Notably, TCA and TCA+ combined with SMOTE-ENN emerge consistently as effective strategies for domain alignment and class balance. LightGBM and Logistic Regression tend to dominate in datasets with extreme class imbalance (e.g., Log4j and Xalan), while Random Forest shows consistent stability in more balanced datasets such as Synapse.

Table 3. Top-Performing Configurations Across Projects Based on Composite Score (calculated as the normalized mean of F1, Recall, AUC, and MCC)

Model	Domain Adaptation	Feature Selection	Resampling	F1	Recall	AUC	MCC	Composite Score
Gradient Boosting	CORAL	None	SMOTE-ENN	0.8	0.72	0.82	0.46	0.91
Gradient Boosting	CORAL	RFE	SMOTE-ENN	0.8	0.72	0.82	0.46	0.91
Extra Trees	Baseline	RFE	SMOTE-ENN	0.81	0.75	0.78	0.42	0.89
Extra Trees	Baseline	None	SMOTE-ENN	0.81	0.75	0.78	0.42	0.89
Random Forest	CORAL	None	SMOTE-ENN	0.8	0.74	0.8	0.39	0.87

Source: (Research result, 2025)

To examine variation and stability across datasets, Tables 4 to 7 present the top five configurations per target project, ranked by composite score. These allow further analysis of which classifiers and preprocessing combinations tend to generalize well in specific contexts.

Table 4. Top Configurations for Camel Dataset Ranked by Composite Score

Model	Domain Adaptation	Feature Selection	Resampling	F1	Recall	AUC	MCC	Composite Score
Logistic Regression	CORAL	SelectKBest	SMOTEENN	0.67	0.55	0.69	0.28	0.73
Logistic Regression	TCA+	SelectKBest	SMOTEENN	0.73	0.67	0.67	0.19	0.72
Logistic Regression	Baseline	SelectKBest	SMOTEENN	0.68	0.56	0.68	0.26	0.72
Logistic Regression	Baseline	RFE	SMOTEENN	0.68	0.56	0.64	0.26	0.71
Logistic Regression	Baseline	None	SMOTEENN	0.68	0.56	0.64	0.26	0.71

Source: (Research result, 2025)

Table 4 lists the highest-ranking configurations for the Camel dataset based on the composite performance score. Logistic Regression combined with SelectKBest or RFE appears frequently among the best configurations. This suggests that simpler models can still perform competitively on Camel when coupled with feature selection and resampling.

Table 5. Top Configurations for Log4j Target Project Based on Composite Performance Score

Model	Domain Adaptation	Feature Selection	Resampling	F1	Recall	AUC	MCC	Composite Score
LightGBM	TCA	None	SMOTEENN	0.87	0.89	0.8	0.46	0.93
Random Forest	TCA	RFE	SMOTEENN	0.87	0.92	0.77	0.43	0.9
Random Forest	TCA	None	SMOTEENN	0.85	0.92	0.79	0.32	0.84
LightGBM	TCA	RFE	SMOTEENN	0.85	0.9	0.78	0.32	0.84
Gradient Boosting	TCA	SelectKBest	None	0.86	0.93	0.77	0.31	0.83

Source: (Research result, 2025)

Table 5 lists the five best model configurations for predicting defects in the Log4j dataset. LightGBM and Random Forest with TCA and SMOTE-ENN dominate the top results, highlighting their strong performance on datasets with severe imbalance like Log4j.

Table 6. Highest-Scoring Configurations for Synapse Dataset Using Composite Metric

Model	Domain Adaptation	Feature Selection	Resampling	F1	Recall	AUC	MCC	Composite Score
Gradient Boosting	CORAL	None	SMOTEENN	0.8	0.72	0.82	0.46	0.91
Gradient Boosting	CORAL	RFE	SMOTEENN	0.8	0.72	0.82	0.46	0.91
Extra Trees	Baseline	RFE	SMOTEENN	0.81	0.75	0.78	0.42	0.89
Extra Trees	Baseline	None	SMOTEENN	0.81	0.75	0.78	0.42	0.89
Random Forest	CORAL	None	SMOTEENN	0.80	0.74	0.8	0.39	0.87

Source: (Research result, 2025)

Table 6 presents the top five configurations tested on the Synapse project. Gradient Boosting and Extra Trees consistently appear at the top, with CORAL and SMOTE-ENN offering improved domain alignment and class balance. This reinforces the importance of domain adaptation in more balanced datasets.

Table 7. Top Model and Preprocessing Combinations for Xalan Project Ranked by Composite Score

Model	Domain Adaptation	Feature Selection	Resampling	F1	Recall	AUC	MCC	Composite Score
LightGBM	TCA+	None	SMOTEENN	0.86	0.89	0.77	0.42	0.90
LightGBM	TCA+	RFE	SMOTEENN	0.86	0.89	0.77	0.42	0.90
Logistic Regression	Baseline	SelectKBest	SMOTEENN	0.82	0.78	0.80	0.38	0.88
Gradient Boosting	TCA+	None	SMOTEENN	0.85	0.87	0.75	0.37	0.87
Gradient Boosting	TCA+	RFE	SMOTEENN	0.85	0.87	0.75	0.37	0.87

Source: (Research result, 2025)

Table 7 showcases the configurations achieving the highest overall scores when applied to the Xalan dataset. LightGBM and Gradient Boosting combined with TCA+ and SMOTE-ENN appear frequently, indicating their robustness in handling domain shift and class imbalance. The composition of top-performing pipelines across all datasets emphasizes the contextual nature of configuration effectiveness. While ensemble-based classifiers such as LightGBM and Gradient Boosting often dominate, traditional models like Logistic Regression can perform competitively when supported by appropriate feature selection and domain adaptation. These results reinforce the study's core contribution that metric-based and computationally lightweight CPDP pipelines can achieve robust performance across heterogeneous software projects. Overall, the variation across datasets emphasizes that CPDP performance is highly sensitive to dataset characteristics and the alignment between source and target distributions validating the necessity of flexible and modular pipelines.

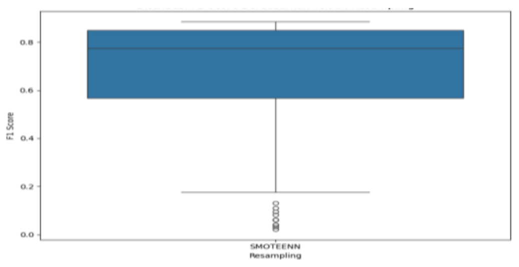
2. Component Analysis Pipeline

TCA and TCA+ were most effective at reducing domain shift, especially on dataset pairs with high distributional differences. Baseline and CORAL produced fluctuating performance. SelectKBest showed stable and superior performance when used with tree-based models and numerical data. RFE excels on Logistic Regression but is more sensitive to noise and high dimensionality. SMOTEENN significantly improves Recall and G-Mean on imbalanced datasets such as Camel and Synapse. However, its impact on MCC and AUC is not always positive, signaling a trade-off between sensitivity and specificity. LightGBM excelled in the number of best configurations overall, indicating generalization ability across domains. Logistic Regression yields the highest MCC values on a given project, suggesting that classical models remain relevant in the CPDP.

The consistently strong performance of configurations involving TCA and SMOTEENN can be attributed to their complementary nature: TCA reduces distributional shift between source and target domains, facilitating better generalization, while SMOTEENN improves recall by addressing class imbalance through oversampling and noise reduction. Feature selection methods such as SelectKBest enhanced model focus on the most predictive attributes, particularly in high-dimensional spaces like Camel. Ensemble classifiers like LightGBM and Gradient Boosting benefit from their ability to capture nonlinear patterns and resist overfitting, especially when paired with TCA. In contrast, Logistic Regression performed best when dimensionality was reduced via feature selection, as it is more sensitive to irrelevant features and domain variance.

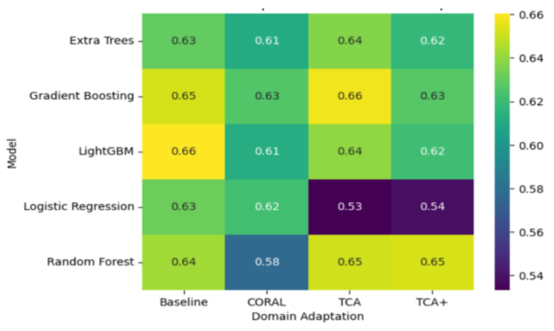
3. Visualization of Performance Distribution

To complement the tabular results, several visualizations were employed to highlight patterns in model behavior, metric relationships, and resampling outcomes across configurations. These figures provide deeper insights into model performance variation across techniques and datasets.



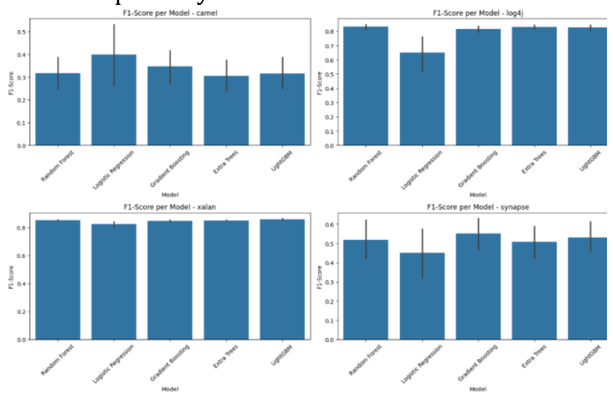
Source: (Research result, 2025)
Figure 2. F1-score distribution for configurations with SMOTEENN resampling.

Figure 2 presents a boxplot of F1-scores for all configurations using SMOTEENN. The results indicate that the majority of models achieve F1-scores above 0.75, with minimal variance, showing the consistency and reliability of SMOTE-ENN. However, a few outliers suggest sensitivity when paired with suboptimal model or domain adaptation choices.



Source: (Research result, 2025)
Figure 3. Average F1-Score per Model vs. Adaptation Domain

Figure 3 visualizes how each classifier performs under different domain adaptation settings. LightGBM and Gradient Boosting show high F1-scores across Baseline, TCA, and CORAL, confirming their resilience to domain shift. Meanwhile, Logistic Regression significantly underperforms with TCA and TCA+, indicating lower adaptability in those scenarios.

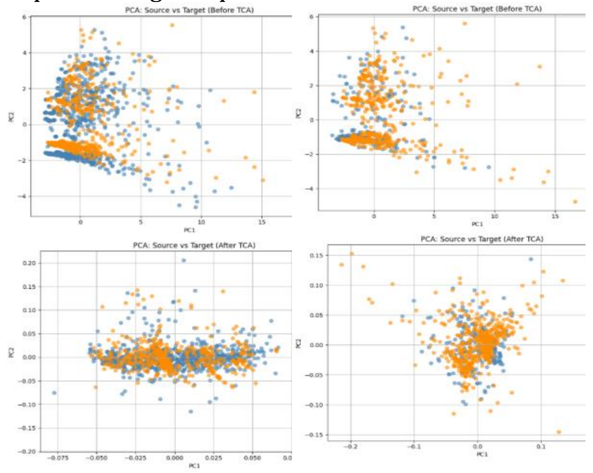


Source: (Research result, 2025)
Figure 4. F1 per Model for each Dataset

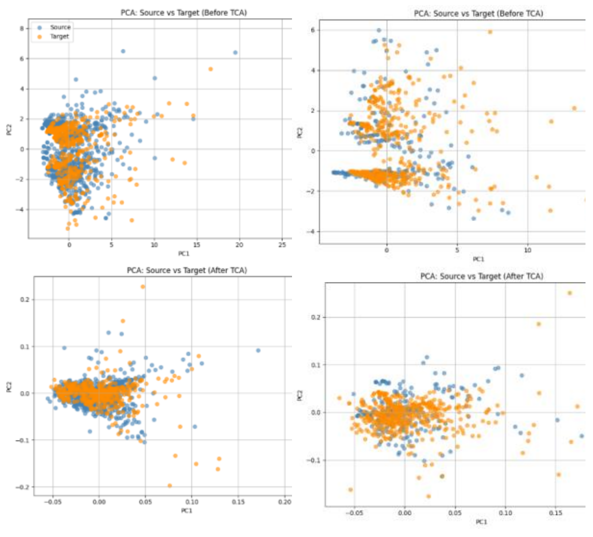
The four subplots in Figure 4 display per-model F1 performance for each target dataset. LightGBM and Extra Trees maintain strong and consistent performance across datasets, while Logistic Regression and Random Forest exhibit greater variability, particularly in Xalan and Camel. These findings reinforce the need for context-specific model tuning in CPDP.

4. Visualization of Domain Distribution Projection PCA

Understanding domain shift is crucial in CPDP. Figure 5 employs PCA to visualize feature distributions of the source and target projects before and after applying Transfer Component Analysis (TCA). This allows us to assess the effectiveness of domain adaptation visually, complementing the quantitative evaluation.



(a) Xalan Dataset (b) Log4j Dataset

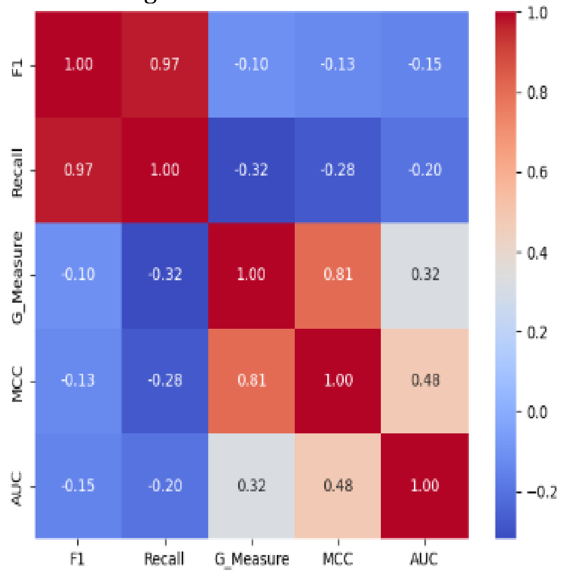


(c) Camel Dataset (d) Synapse Dataset

Source: (Research result, 2025)
Figure 5. PCA visualization of source–target domains before and after TCA, illustrating that TCA effectively reduces distributional shift and explains the observed improvements in CPDP performance.

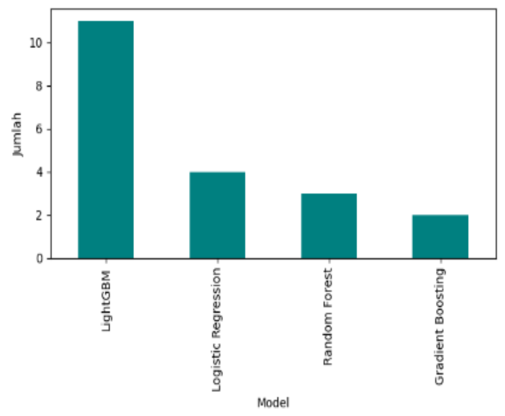
Each subfigure 5(a–d) corresponds to a dataset. In the pre-TCA plots, there is a clear separation between source and target domains, implying strong domain divergence: Xalan 5(a): The source data clusters densely in one region, while the target spreads more widely. Log4j 5(b): Shows even more separation, especially along PC1. Camel and Synapse 5(c, d): Source and target clusters are nearly disjoint. In contrast, the post-TCA projections demonstrate that the domains become much more aligned, indicated by overlapping clusters in PC1-PC2 space. This visual confirmation explains the performance improvements of TCA in the previous tables (e.g., Tables 5 and 7), where it significantly contributed to higher composite scores. These projections clearly show that TCA effectively reduces distributional shift, a key challenge in CPDP. This supports the empirical results where configurations with TCA or TCA+ achieved superior performance.

To complement the insights from domain alignment, the subsequent figures (Figures 6–8) explore the interrelation between evaluation metrics and model performance patterns. These visualizations help explain why certain models and configurations consistently achieve superior results in CPDP settings.



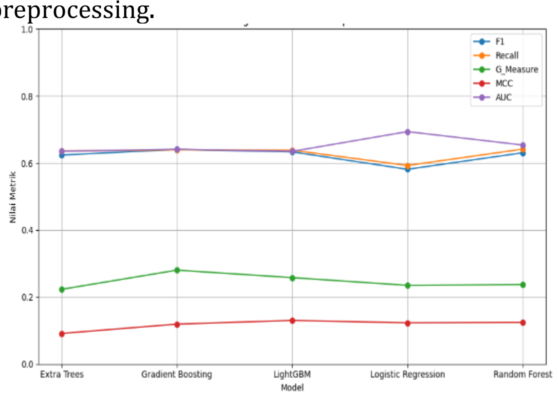
Source: (Research result, 2025)
Figure 6. Correlation between Evaluation Metrics

Figure 6 shows a correlation heatmap among the evaluation metrics. F1, Recall, and G-Mean are strongly correlated (0.97–1.00), which implies that models optimizing one of them tend to improve others. Interestingly, AUC and MCC have lower or negative correlation with these metrics, highlighting their independent behavior and importance for multi-perspective evaluation.



Source: (Research result, 2025)
Figure 7. Number of Model Appearances in Top-20 Based on F1-Score

This bar chart on Figur 7 shows the frequency of each model in the top 20 configurations based on F1-score. LightGBM dominates, reflecting its robustness and generalization across different target projects. Logistic Regression, despite being simple, also appears frequently — confirming its continued relevance in CPDP when paired with proper preprocessing.



Source: (Research result, 2025)
Figure 8. Comparison of Average Evaluation Metrics per Model

The final comparison in Figure 8 plots average metric values per model. LightGBM leads overall, while Gradient Boosting shows stronger G-Measure and Logistic Regression ranks best in MCC. This indicates that no single model dominates across all metrics, emphasizing the importance of multi-metric evaluation in defect prediction.

These results confirm that traditional pipeline-based CPDP can provide competitive prediction performance against complex AST-based or deep learning approaches. The integration of domain adaptation TCA, SMOTEENN resampling, SelectKBest feature selection, and models such as LightGBM can be a practical solution for organizations with computational limitations. The effectiveness of TCA in aligning feature distributions across projects is visually and quantitatively proven, while the dominance of LightGBM in the best configuration supports the claim that boosting models have high domain adaptability.

5. Performance Comparison with TLSTM

To further contextualize the performance of the proposed pipeline, Table 7 compares our result on the Camel→Xerces configuration with a recent deep learning-based approach (TLSTM) introduced by (Tao et al., 2024).

Table 8. Summary Comparison of Proposed Model and Deep Learning-based CPDP (with statistical significance testing)

Model	Approach Type	Feature Type	Technique	F1 (Camel→Xerces)
This Study	Traditional ML + DA (Logistic Regression)	Metric-based	TCA+ + SMOTEENN + SelectKBest	0.73
TLSTM	Deep Learning (LSTM)	Semantic-based (AST)	TLSTM (AST + Transfer Learning)	0.747

Source: (Research result, 2025; (Tao et al., 2024)

While TLSTM achieves a slightly higher F1-score of 0.747, our traditional metric-based pipeline yields a close score of 0.730 using Logistic Regression combined with TCA+, SMOTEENN, and SelectKBest. Although formal statistical significance testing was not conducted, these results indicate that the proposed approach can achieve performance that is practically competitive with TLSTM, while offering lower computational cost and simpler implementation.

CONCLUSION

This research demonstrates that traditional metric-based CPDP pipelines combining domain adaptation, feature selection, resampling, and classical machine learning models can still deliver competitive prediction performance, particularly in new software projects where historical defect data is unavailable and domain shift is prominent. Among the 120 combinations tested, techniques such as TCA and TCA+ proved effective in aligning feature distributions across domains, contributing to notable improvements in F1-Score, Recall, and G-Mean. The SMOTEENN resampling method also improved the detection of minority (buggy) instances without destabilizing other performance metrics. LightGBM and Logistic Regression emerged as the most consistent and reliable models, with the best configuration achieving a composite score of up to 0.812 on the Log4j project. PCA visualizations further support the effectiveness of domain alignment, while correlation analysis confirmed the value of using multiple evaluation metrics to ensure unbiased assessment.

The primary contribution of this study lies in offering a lightweight, reproducible, and easily deployable CPDP solution that does not depend on deep learning or resource-intensive processing. This makes the proposed pipeline highly applicable in industrial contexts, particularly for small to medium software teams or organizations operating under limited infrastructure or tight release cycles. In practical terms, this pipeline can be integrated into existing CI/CD workflows, used as a pre-testing quality gate, or embedded in automated QA systems to prioritize modules for further testing or code review. Its low complexity enables easier debugging and transparency, which are often critical in industrial settings.

Future work may explore hybrid pipelines that combine lightweight semantic features with metric-based learning, validation on large-scale industrial datasets, and the integration of AutoML and explainability modules to further enhance adoption, scalability, and trustworthiness in real-world software engineering environments..

REFERENCE

Albattah, W., & Alzahrani, M. (2024). Software Defect Prediction Based on Machine Learning and Deep Learning Techniques: An Empirical Approach. *AI (Switzerland)*, *5*(4), 1743–1758. https://doi.org/10.3390/ai5040086

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1). https://doi.org/10.1186/s12864-019-6413-7

Farid, A. B., Fathy, E. M., Eldin, A. S., & Abd-Elmegid, L. A. (2021). Software defect prediction using hybrid model (CBIL) of convolutional neural network (CNN) and bidirectional long short-term memory (Bi-LSTM). *PeerJ Computer Science*, *7*, 1–22. https://doi.org/10.7717/peerj-cs.739

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

Haldar, S., & Fernando Capretz, L. (2024). Feature Importance in the Context of Traditional and Just-In-Time Software Defect Prediction Models. In *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*.

Kumar, P. H., & Bhat, S. (2024). Enhancing Regional Plagiarism Detection Using a Backtrack Matching Model: A Precision, Recall, and F1 Score-Based Evaluation. In *Journal of Information Systems Engineering and Management* (Vol. 2025). Retrieved from https://www.jisem-journal.com/

Kumar, P. S., Nayak, J., & Behera, H. S. (2022). Model-based Software Defect Prediction from Software Quality Characterized Code Features by using Stacking Ensemble Learning. *Journal of Engineering Science and Technology Review*, *15*(2), 137–155. https://doi.org/10.25103/jestr.152.17

Ren, J., Peng, C., Zheng, S., Zou, H., & Gao, S. (2022). An Approach to Improving Homogeneous Cross-Project Defect Prediction by Jensen-Shannon Divergence and Relative Density. *Scientific Programming*, *2022*. https://doi.org/10.1155/2022/4648468

Sharma, U., & Sadam, R. (2022). *An Empirical Evaluation of Defect Prediction Models Using Project-Specific Measures*. Retrieved from http://ceur-ws.org

Song, H., Pan, Y., Guo, F., Zhang, X., Ma, L., & Jiang, S. (2024). ConCPDP: A Cross-Project Defect Prediction Method Integrating Contrastive Pretraining and Category Boundary Adjustment. *IET Software*, *2024*(1). https://doi.org/10.1049/2024/5102699

Sotto-Mayor, B., & Kalech, M. (2024). A Survey on Transfer Learning for Cross-Project Defect Prediction. *IEEE Access*, *12*, 93398–93425. https://doi.org/10.1109/ACCESS.2024.3424311

Stradowski, S., & Madeyski, L. (2024). Costs and Benefits of Machine Learning Software Defect Prediction: Industrial Case Study. *FSE Companion - Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, 92–103. Association for Computing Machinery, Inc. https://doi.org/10.1145/3663529.3663831

Tao, H., Fu, L., Cao, Q., Niu, X., Chen, H., Shang, S., & Xian, Y. (2024). Cross-Project Defect Prediction Using Transfer Learning with Long Short-Term Memory Networks. *IET Software*, *2024*(1). https://doi.org/10.1049/2024/5550801

Tong, H., Liu, B., Wang, S., & Li, Q. (2019). *Transfer-Learning Oriented Class Imbalance Learning for Cross-Project Defect Prediction*. Retrieved from http://arxiv.org/abs/1901.08429

Vescan, A., Găceanu, R., & Şerban, C. (2024). Exploring the impact of data preprocessing techniques on composite classifier algorithms in cross-project defect prediction. *Automated Software Engineering*, *31*(2). https://doi.org/10.1007/s10515-024-00454-9

Yang, C., Fan, Z., Wu, J., Zhang, J., Zhang, W., Yang, J., & Yang, J. (2021). The Diagnostic Value of Soluble ST2 in Heart Failure: A Meta-Analysis. *Frontiers in Cardiovascular Medicine*, Vol. 8. Frontiers Media SA. https://doi.org/10.3389/fcvm.2021.685904

Zimmermann, T., Nagappan, N., Gall, H., Giger, E., & Murphy, B. (2009). Cross-project defect prediction. *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, 91–100. New York, NY, USA: ACM. https://doi.org/10.1145/1595696.1595713