MUST: ***Journal of Mathematics Education, Science and Technology***
Vol. 10, No. 1, Juli 2025 Hal 58-67
DOI: http://doi.org/10.30651/must.v10i1.26548

ENSEMBLE AND VOTING APPROACHES FOR DEFECT PREDICTION ACROSS MULTIPLE SOFTWARE PROJECTS

Kirso[1], Agus Subekti[2]
[1, 2]Universitas Nusa Mandiri
14230033@nusamandiri.ac.id[1], agus@nusamandiri.ac.id[2].

Received 31 Mei 2025; revised 24 Juni 2025; accepted 26 Juni 2025.

ABSTRAK

Penelitian ini melakukan eksperimen dengan metode ensemble, tuning hyperparameter, dan voting untuk meningkatkan prediksi cacat perangkat lunak pada berbagai proyek menggunakan dataset Kamei. Dengan aplikasi lima model machine learning LightGBM, XGBoost, Random Forest, Extra Trees, dan Gradient Boosting pada enam proyek: Bugzilla, Columba, JDT, Mozilla, Platform, dan Postgres. Hasilnya secara umum, model menunjukkan performa yang baik saat diuji pada dataset proyek dengan karakteristik serupa atau hubungan yang kuat, seperti pada proyek Mozilla, JDT, dan Platform, dengan akurasi dan skor F1 di atas 80%. Hal ini mengindikasikan bahwa pola cacat yang dipelajari dari satu proyek dapat diterapkan secara efektif pada proyek lain yang serupa. Namun, performa model menurun secara signifikan saat memprediksi proyek Bugzilla dari proyek lain, yang menunjukkan adanya perbedaan pola cacat atau ketidakcocokan fitur yang cukup besar. Perbedaan distribusi data antar proyek menjadi tantangan utama dalam CPDP. Oleh karena itu, dibutuhkan teknik adaptasi domain atau transformasi fitur untuk mengurangi perbedaan antar proyek sehingga model dapat mengenali pola cacat dengan lebih baik di berbagai proyek. Meskipun ada peningkatan, perbedaan data antar proyek dan ketidakseimbangan kelas masih membatasi kinerja prediksi. Penelitian selanjutnya perlu mengatasi tantangan ini.

Kata kunci: Cross Project Defect Prediction (CPDP), Ensemble Learning, Hyperparameter Tuning, Kamei Dataset, Voting

ABSTRACT

This study conducted experiments using ensemble methods, hyperparameter tuning, and voting to improve software defect prediction across multiple projects using the Kamei dataset. Five machine learning models LightGBM, XGBoost, Random Forest, Extra Trees, and Gradient Boosting were applied to six projects: Bugzilla, Columba, JDT, Mozilla, Platform, and Postgres. Overall, the models demonstrated good performance when tested on datasets of projects with similar

characteristics or strong relationships, such as Mozilla, JDT, and Platform, achieving accuracy and F1 scores above 80%. This indicates that defect patterns learned from one project can be effectively applied to similar projects. However, the models' performance dropped significantly when predicting defects in the Bugzilla project from other projects, indicating notable differences in defect patterns or feature incompatibility. Differences in data distribution across projects remain a major challenge in CPDP. Therefore, domain adaptation techniques or feature transformation methods are needed to reduce inter-project differences, enabling the models to better recognize defect patterns across projects. Despite some improvements, data differences and class imbalance still limit prediction performance. Future research should address these challenges.

Keywords: Cross Project Defect Prediction (CPDP), Ensemble Learning, Hyperparameter Tuning, Kamei Dataset, Voting

INTRODUCTION

Cross project defect prediction (CPDP) has emerged as a critical area within software engineering, offering a promising approach for enhancing software quality and maintainability by leveraging data from multiple software projects. This methodology utilizes historical defect data from source projects to predict potential defects in new, target projects, thereby supporting a more efficient software development lifecycle (Bala et al., 2023; Lei et al., 2024)

While within project defect prediction (WPDP) methods rely on historical data from a single project, they often suffer from data scarcity particularly in the early stages of development. CPDP addresses this limitation by enabling the reuse of defect data across different projects (Goel et al., 2022; Li et al., 2025). However, despite its potential, CPDP still faces challenges in ensuring model generalizability and handling heterogeneity across projects. Compared to previous studies, which primarily focused on improving prediction accuracy within isolated project environments, this study emphasizes the development and evaluation of more robust cross project learning strategies to bridge these gaps and enhance defect prediction performance in data scarce target projects.

Innovative techniques have been explored to enhance CPDP outcomes. These techniques include novel hybrid models that combine classical machine learning methods, such as Support Vector Machines and Random Forests, with deep learning approaches, leading to improved prediction capabilities (Kumar

& Saxena, 2024). Additionally, the issue of class imbalance a prevalent challenge in CPDP has prompted researchers to devise specialized frameworks aimed at enhancing the learning process and optimizing feature selection (Sekaran & Lawrence, 2025; Tahir et al., 2023). Focusing on comprehensive feature representation and managing distributional variations is critical for ensuring that CPDP methods are both robust and reliable (Bala et al., 2023; Gul et al., 2023). Machine learning techniques play a crucial role in developing CPDP methodologies. By leveraging complex algorithms ranging from various classical models to advanced hybrid frameworks that incorporate deep learning researchers aim to improve prediction accuracy and address challenges such as data imbalance and feature heterogeneity between source and target projects (Bhat & Farooq, 2021; Haque et al., 2024; Zhao et al., 2022).

This study employs various approaches, such as multiple machine learning classifiers combined with parameter tuning and ensemble learning, to perform Cross-Project Defect Prediction (CPDP). The ensemble learning method used in this study is voting. The objective of this research is to conduct CPDP by applying hyperparameter tuning and ensemble learning using the PyCaret framework.

RESEARCH METHOD

This research was conducted by analyzing previously published datasets, specifically utilizing the Kamei Dataset, which includes several open-source software projects such as Bugzilla, Columba, JDT, Mozilla, Platform, and Postgres. The dataset is referenced from the study by (Chen et al., 2022) and is publicly available at: https://github.com/Kirso098/CPDP-Kamei. The research methodology comprises three primary stages: data acquisition, model development, and model evaluation. In the data acquisition phase, historical data from the Kamei dataset is systematically gathered and prepared for analysis. During the model development phase, each project dataset is utilized for training and testing various machine learning algorithms, including LightGBM, XGBoost, Random Forest, Gradient Boosting, Extra Trees, AdaBoost, Decision Tree, K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), Ridge Classifier, Logistic Regression, Support Vector Machine (SVM), Naive Bayes, and Quadratic Discriminant Analysis (QDA). The final stage involves model evaluation, which is conducted

under a cross-project defect prediction (CPDP) framework, wherein the model is trained on one project and tested on different projects to assess generalizability. The performance of each model is evaluated using standard classification metrics, including Accuracy, Precision, Recall, F1-Score, and the Area Under the Receiver Operating Characteristic Curve (AUC-ROC).

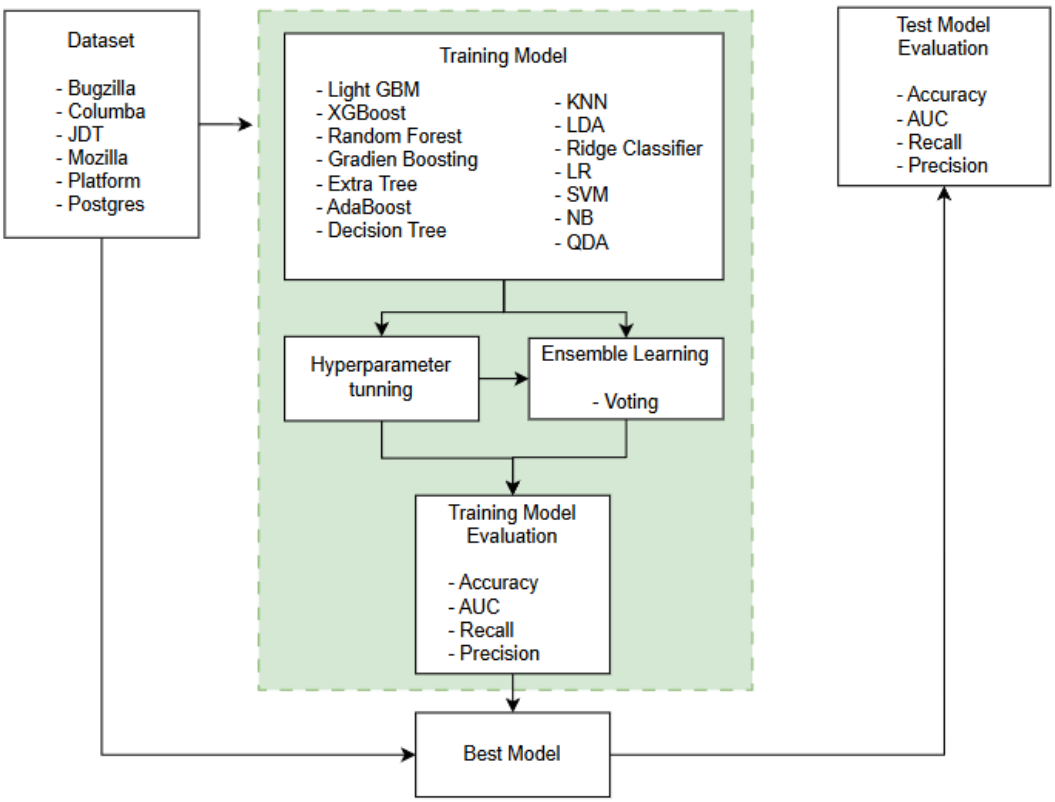

Figure 1. Research Method

Next, the top three algorithms with the highest accuracy will be selected for hyperparameter tuning. The best results from each algorithm, both before and after tuning, will then be combined using the voting method (ensemble learning). Among all the algorithms, the one with the best accuracy will be selected as the final model.

RESEARCH RESULTS AND DISCUSSIONS

Table 1. Bug Ratio Dataset

No	Dataset Name	Total Data	Attributes	Non Bug	Bug	Bug Ratio
1	Bugzilla	4620	15	2924	1696	36,71%
2	Columba	4455	15	3094	1361	30,55%
3	JDT	4620	15	30297	5089	14,38%
4	Mozilla	98275	15	93126	5149	5,24%

5	Platform	64250	15	54798	9452	14,71%
6	Postgres	20431	15	15312	5119	25,06%

The table above presents the details of the six datasets used in this study: Bugzilla, Columba, JDT, Mozilla, Platform, and Postgres, all of which are part of the Kamei Dataset. Each dataset contains 15 attributes, with varying numbers of instances, ranging from 4,455 entries (Columba) to 98,275 entries (Mozilla). In terms of data distribution between the Bug and Non-Bug classes, there is a significant imbalance in several datasets. For example, the Mozilla dataset has a very low bug ratio of only 5.24%, while Bugzilla has the highest bug ratio at 36.71%. This indicates that most datasets face challenges related to imbalanced classification, which must be addressed during the predictive model training process. The JDT and Platform datasets have relatively large numbers of bug instances (5,089 and 9,452, respectively), but due to the overall size of the datasets, the bug ratios remain low. This highlights the heterogeneity among the datasets in terms of both size and class distribution, making Cross Project Defect Prediction (CPDP) experiments particularly relevant and necessary.

Here is the result of the experiment conducted in this study, which involved six training models using cross dataset testing.

Table 2. CPDP With Bugzilla Model

No	Model – Data Uji	Accuracy	Precision	Recall	F1-Score
1	Bugzilla - Columba	0.69	0.66	0.69	0.67
2	Bugzilla - JDT	0.80	0.80	0.80	0.80
3	Bugzilla - Mozilla	0.80	0.93	0.80	0.85
4	Bugzilla - Platform	0.80	0.81	0.80	0.80
5	Bugzilla - Postgres	0.74	0.71	0.74	0.72

Based on Table 2, the results of model testing using the Bugzilla model show that the model performs fairly well overall. The model performs the lowest on the Columba dataset with an accuracy of 69%, indicating that the model has some difficulty in recognizing patterns within this data. Conversely, for the JDT, Mozilla, and Platform datasets, the model demonstrates excellent performance, with accuracy, precision, recall, and F1-score consistently around 80%. Notably, for the Mozilla dataset, the precision reaches 93%, meaning the model is highly accurate in predicting positive cases and rarely makes errors. Meanwhile, the

Postgres dataset shows a decent performance, though not as strong as the three datasets mentioned earlier.

Table 3. CPDP With Columba Model

No	Model – Data Uji	Accuracy	Precision	Recall	F1-Score
1	Columba - Bugzilla	0.69	0.69	0.69	0.65
2	Columba - JDT	0.81	0.81	0.81	0.81
3	Columba - Mozilla	0.85	0.93	0.85	0.88
4	Columba - Platform	0.81	0.80	0.81	0.81
5	Columba - Postgres	0.78	0.75	0.78	0.75

Based on the results in Table 3, the evaluation of the Columba model on five test datasets shows generally good overall performance. The model achieved its lowest results when tested on the Bugzilla dataset, with accuracy, precision, and recall at 69%, and an F1-score of 0.65. This indicates that the model was not optimal in recognizing patterns within that dataset. However, the model performed very well on the JDT, Platform, and especially Mozilla datasets. On the Mozilla dataset, the model achieved 85% accuracy, the highest F1-score of 0.88, and 93% precision, indicating a high level of accuracy in predicting positive instances. Meanwhile, on the Postgres dataset, the model still showed good performance, with 78% accuracy and an F1-score of 0.75.

Table 4. CPDP With JDT Model

No	Model – Data Uji	Accuracy	Precision	Recall	F1-Score
1	JDT - Bugzilla	0.67	0.69	0.67	0.61
2	JDT - Columba	0.73	0.73	0.73	0.68
3	JDT - Mozilla	0.87	0.93	0.87	0.89
4	JDT - Platform	0.84	0.82	0.84	0.83
5	JDT - Postgres	0.80	0.78	0.80	0.77

Based on the results in Table 4, the evaluation of the JDT model on five test datasets shows that the model has quite varied performance. The lowest performance was observed on the Bugzilla dataset, with only 67% accuracy and an F1-score of 0.61, indicating that the model still struggles to recognize patterns in this dataset. The model's performance improved when tested on the Columba dataset, achieving 73% accuracy and an F1-score of 0.68. The best performance was achieved on the Mozilla dataset, where the model reached 87% accuracy, 93% precision, 87% recall, and the highest F1-score of 0.89. Excellent results were also seen on the Platform and Postgres datasets, with accuracies of 84% and 80%,

respectively. Overall, the JDT model performs very well, especially on the Mozilla and Platform datasets, but still requires improvement when applied to the Bugzilla dataset to ensure greater stability and generalization.

Table 5. CPDP With Mozilla Model

No	Model – Data Uji	Accuracy	Precision	Recall	F1-Score
1	Mozilla - Bugzilla	0.63	0.56	0.63	0.49
2	Mozilla - Columba	0.70	0.79	0.70	0.57
3	Mozilla - JDT	0.86	0.81	0.86	0.79
4	Mozilla - Platform	0.85	0.84	0.85	0.79
5	Mozilla - Postgres	0.75	0.81	0.75	0.65

Based on the results in Table 5, the evaluation of the Mozilla model on five test datasets produced varied outcomes. The lowest performance occurred when tested on the Bugzilla dataset, with 63% accuracy and an F1-score of 0.49, indicating that the model had difficulty recognizing patterns in this dataset. Performance improved when tested on the Columba dataset, achieving 70% accuracy and an F1-score of 0.57. The best results were obtained when the Mozilla model was tested on the JDT and Platform datasets, with accuracies of 86% and 85%, respectively, and identical F1-scores of 0.79. This indicates that the Mozilla model was able to understand the patterns in the JDT and Platform datasets very well. Meanwhile, on the Postgres dataset, performance remained good, with 75% accuracy and an F1-score of 0.65. Overall, the Mozilla model performed well, especially when tested on the JDT and Platform datasets, although improvements are still needed for the Bugzilla dataset.

Table 6. CPDP With Platform Model

No	Model – Data Uji	Accuracy	Precision	Recall	F1-Score
1	Platform - Bugzilla	0.69	0.72	0.69	0.62
2	Platform - Columba	0.72	0.72	0.70	0.67
3	Platform - JDT	0.86	0.82	0.86	0.82
4	Platform - Mozilla	0.90	0.93	0.90	0.92
5	Platform - Postgres	0.80	0.79	0.80	0.76

Based on the model Platform's testing on five test datasets, it is evident that the model's performance is quite stable and robust. The lowest performance occurred when tested on the Bugzilla data, with an accuracy of 69% and an F1-score of 0.62, indicating results that still need improvement. Performance improved on the Columba data, with an accuracy of 72% and an F1-score of 0.67. The model showed very good results when tested with the JDT and Mozilla data, achieving

accuracies of 86% and 90%, and F1-scores of 0.82 and 0.92. The highest values among all tests. Meanwhile, on the Postgres data, the model also demonstrated fairly good performance with an accuracy of 80% and an F1-score of 0.76. Overall, the Platform model performed consistently and excellently, especially when tested with Mozilla and JDT data.

Table 7. CPDP With Postgres Model

No	Model – Data Uji	Accuracy	Precision	Recall	F1-Score
1	Postgres - Bugzilla	0.66	0.66	0.66	0.58
2	Postgres - Columba	0.73	0.73	0.73	0.73
3	Postgres - JDT	0.82	0.82	0.82	0.82
4	Postgres - Mozilla	0.89	0.93	0.89	0.91
5	Postgres - Platform	0.82	0.82	0.82	0.82

Based on the testing results of the Postgres model, its overall performance appears to be quite good. The lowest values occurred when tested with the Bugzilla data, with an accuracy of 66% and an F1-score of 0.58, indicating relatively low performance on that dataset. The results improved when tested with the Columba data (accuracy and F1-score: 0.73). The model demonstrated high and consistent performance when tested on the JDT and Platform data, both recording an accuracy and F1-score of 0.82. The best performance was shown when tested on the Mozilla data, with an accuracy of 89% and the highest F1-score of 0.91. Overall, the Postgres model exhibits very good performance, especially on the Mozilla data, and remains stable on other datasets.

CONCLUSIONS

This study shows that the performance of models in Cross-Project Defect Prediction (CPDP) is highly influenced by the similarity between the source and target projects. The models perform well when applied to projects with similar characteristics, such as Mozilla, JDT, and Platform, achieving accuracy and F1-scores above 80%. This indicates that defect patterns learned from one project can be used effectively to predict defects in other similar projects. On the other hand, performance drops significantly when predicting defects in the Bugzilla project, likely due to differences in defect patterns or feature incompatibility.

Theoretically, this finding highlights the importance of project similarity to improve model accuracy and generalization in CPDP. Practically, it suggests that prediction models should be trained on data from projects that are structurally similar to the target project to ensure more reliable predictions. This study still has several limitations, such as data distribution differences and class imbalance across projects, which ensemble methods alone could not fully resolve. It also does not include deep learning or transfer learning approaches, which may offer better feature abstraction. In addition, the dataset used was limited to several open-source projects and may not represent all real-world software environments.

For future research, it is recommended to apply domain adaptation techniques or feature transformation methods to align data distribution across projects. Exploring hybrid models that combine machine learning with deep learning or meta-learning may also improve CPDP performance. Additionally, expanding the dataset to include more diverse projects and incorporating temporal or semantic information of code may provide deeper insights into defect patterns across projects.

ACKNOWLEDGEMENTS

Thank you to Li-qiong Chen, Can Wang, and Shi-long Song for providing this valuable dataset.

REFFERENCES

Bala, Y. Z., Samat, P. A., Sharif, K. Y., & Manshor, N. (2023). Improving Cross-Project Software Defect Prediction Method Through Transformation and Feature Selection Approach. *IEEE Access*, *11*, 2318–2326. https://doi.org/10.1109/ACCESS.2022.3231456

Bhat, Nayeem Ahmad, & Farooq, Sheikh Umar. (2021). Local modeling approach for cross-project defect prediction. *Intelligent Decision Technologies*, *15*(4), 623–637. https://doi.org/10.3233/IDT-210130

Chen, L., Wang, C., & Song, S. (2022). Software defect prediction based on nested-stacking and heterogeneous feature selection. *Complex & Intelligent Systems*, *8*(4), 3333–3348. https://doi.org/10.1007/s40747-022-00676-y

Goel, L., Nandal, N., & Gupta, S. (2022). An optimized approach for class imbalance problem in heterogeneous cross project defect prediction [version 1; peer review: 1 approved with reservations]. *F1000Research*, *11*(1060). https://doi.org/10.12688/f1000research.123616.1

Gul, S., Faiz, R. Bin, Aljaidi, M., Samara, G., Alsarhan, A., & al-Qerem, A. (2023). Impact Evaluation of Significant Feature Set in Cross Project for Defect Prediction through Hybrid Feature Selection in Multiclass. *BioRxiv*, 2023.07.20.549868. https://doi.org/10.1101/2023.07.20.549868

Haque, R., Ali, A., McClean, S., Cleland, I., & Noppen, J. (2024). Heterogeneous Cross-Project Defect Prediction Using Encoder Networks and Transfer Learning. *IEEE Access*, *12*, 409–419. https://doi.org/10.1109/ACCESS.2023.3343329

Kumar, H., & Saxena, V. (2024). Software Defect Prediction Using Hybrid Machine Learning Techniques: A Comparative StudyNo Title. *Journal of Software Engineering and Applications*, *17*(4), 155–171. https://doi.org/https://doi.org/10.4236/jsea.2024.174009

Lei, T., Xue, J., Man, D., Wang, Y., Li, M., & Kong, Z. (2024). SDP-MTF: A Composite Transfer Learning and Feature Fusion for Cross-Project Software Defect Prediction. In *Electronics* (Vol. 13, Issue 13). https://doi.org/10.3390/electronics13132439

Li, T., Wang, Z., & Shi, P. (2025). Within-project and cross-project defect prediction based on model averaging. *Scientific Reports*, *15*(1), 6390. https://doi.org/10.1038/s41598-025-90832-4

Sekaran, K., & Lawrence, S. P. A. (2025). Leveraging Levy Flight and Greylag Goose Optimization for Enhanced Cross-Project Defect Prediction in Software Evolution. *Journal of Software: Evolution and Process*, *37*(3), e70013. https://doi.org/https://doi.org/10.1002/smr.70013

Tahir, T., Gencel, C., Rasool, G., Umer, T., Rasheed, J., Yeo, S. F., & Cevik, T. (2023). Early Software Defects Density Prediction: Training the International Software Benchmarking Cross Projects Data Using Supervised Learning. *IEEE Access*, *11*, 141965–141986. https://doi.org/10.1109/ACCESS.2023.3339994

Zhao, Y., Zhu, Y., Yu, Q., & Chen, X. (2022). Cross-Project Defect Prediction Considering Multiple Data Distribution Simultaneously. In *Symmetry* (Vol. 14, Issue 2). https://doi.org/10.3390/sym14020401