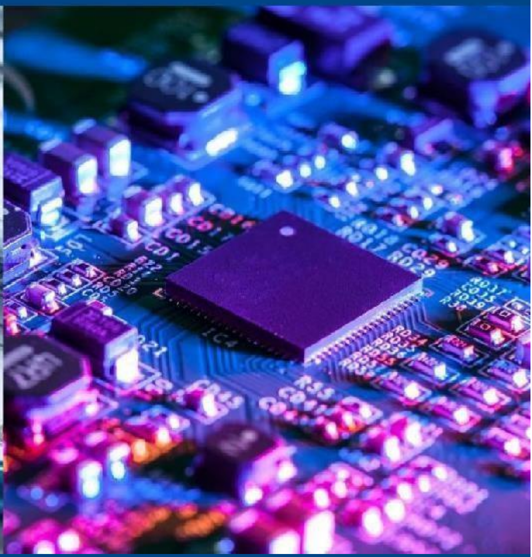
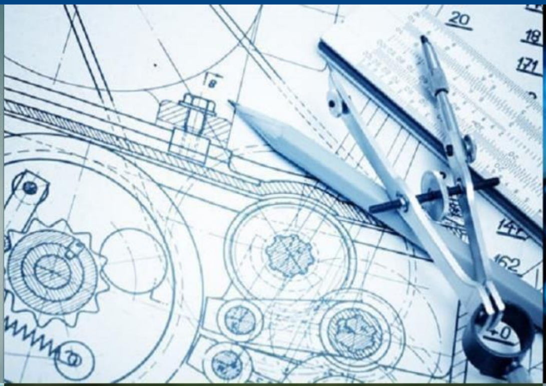
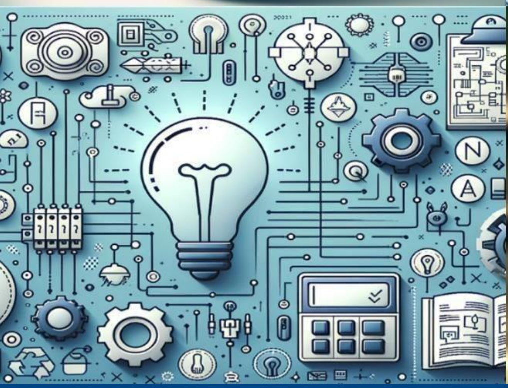
ISSN: 2582-7219



International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)





Impact Factor: 8.206

Volume 8, Issue 8, August 2025

© 2025 IJMRSET | Volume 8, Issue 8, August 2025| DOI:10.15680/IJMRSET.2025.0808074

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

DETECTING APPLICATION DEFECTS USING INTER-PROJECT COMPARISON A REVIEW OF SYSTEM EXTENSION ON CATEGORY INEQUALITY

Gunasekaran K, Janavi P

Assistant Professor, Department of MCA, AMC Engineering College, Bengaluru, India

Student, Department of MCA, AMC Engineering College, Bengaluru, India

ABSTRACT: Software fault prediction is a critical aspect of software engineering aimed at improving software quality and reliability. However, it faces significant challenges, particularly the class imbalance issue in fault data and the necessity for robust predictive models that generalize effectively across different projects. This research explores these challenges through a detailed cross-project analysis and evaluates their impact on software fault prediction. Our study addresses three primary research questions. First, we investigate the problem of class imbalance, where the number of faulty instances is significantly lower than non-faulty ones, making accurate prediction difficult. By conducting extensive experiments using various classifiers on datasets from diverse software projects, we highlight how class imbalance affects model performance and the importance of mitigating it. Second, we assess the reliability of cross-project prediction to understand how well models trained on one project can predict faults in another. Our findings show that models perform better when the training and target datasets share similar characteristics. Third, we analyze the effect of increasing the number of training samples from different projects and demonstrate that doing so enhances prediction accuracy and generalization. We further compare classifier performance using standard evaluation metrics such as accuracy, precision, recall, and F1 score. Our findings emphasize the need to consider both class imbalance and model generalization in developing effective and dependable software fault prediction models. This research contributes valuable insights and guidance for constructing reliable predictive models capable of operating across different software projects.

I. INTRODUCTION

These days in software engineering, making sure our software is high-quality and reliable is more important than ever. As software systems grow increasingly complex and integral to critical applications, identifying and correcting faults before deployment becomes a vital component of the software development life cycle. One of the most effective techniques for improving software quality is Software Fault Prediction (SFP), which aims to detect modules or components that are likely to be defective so that testing and maintenance efforts can be prioritized accordingly.

Traditional fault prediction methods generally rely on within-project data, where historical data from a single software project is used to train predictive models. While these approaches have shown promise, they suffer from a key limitation class imbalance. In most datasets, there are far more non-faulty (majority) instances than faulty (minority) ones.

This skew can lead to biased models that underperform when identifying the minority (fault-prone) class, reducing the overall reliability of predictions. Another critical challenge lies in the generalizability of prediction models. In real-world scenarios, especially in new or evolving software projects, sufficient historical data for training may not be available. This limitation has led to the exploration of Cross-Project Defect Prediction (CPDP), where models are trained on data from other software projects. However, CPDP introduces its own set of challenges, including variations in feature distributions, project characteristics, and metrics, which can negatively impact model performance if not carefully handled.

This project aims to address both of these challenges through a structured investigation into the impact of class imbalance and model generalization in cross-project settings. We apply a range of machine learning classifiers, conduct parameter tuning for optimization, and evaluate performance across multiple open-source datasets using standard metrics such as accuracy, precision, recall, and F1 score. By leveraging data preprocessing techniques, resampling strategies (e.g., SMOTE, ADASYN), and cross-validation, the proposed system aims to build robust models that can effectively generalize across projects. Through this research, we contribute valuable insights into improving the performance of fault prediction systems in real-world software environments, where data availability and quality may vary significantly across projects.

II. LITERATURE SYRVEY

Mende and Giger explored the effectiveness of software metrics in predicting faulty classes. Using industrial data, they analyzed metric correlations with defects and found that specific object-oriented metrics, such as complexity and coupling, are strong predictors. The study concluded that metric-based prediction models can significantly aid early defect detection.

Jureczko and Madeyski studied how clustering software projects could improve defect prediction. They demonstrated that grouping similar projects based on software metrics improves the accuracy of cross-project predictions. Their work highlights the potential of cluster-based modeling in scenarios lacking historical defect data.

Sun et al. proposed a cost-sensitive boosting algorithm to address class imbalance in classification problems. Their method assigns higher weights to minority class instances, improving the detection rate for rare but critical cases such as software defects. Experiments showed superior performance compared to standard boosting techniques.

Chawla et al. They introduced SMOTE, a popular oversampling method used to handle imbalanced datasets.

By generating synthetic minority class examples, SMOTE enhances classifier learning, leading to improved recall in defect prediction tasks. This method became a key approach for dealing with skewed software defect datasets.

He and Wu applied Radial Basis Function (RBF) networks to imbalanced defect datasets. They integrated sampling strategies with RBF learning to improve predictive accuracy. The results indicated that this hybrid approach outperforms traditional methods in detecting faulty modules.

Kaliraj and Jaiswal utilized Generative Adversarial Networks (GANs) to synthesize realistic minority class data for defect prediction. This approach mitigates imbalance while preserving data distribution, leading to significant improvements in precision and recall for fault-prone modules.

Manchala and Bisi (2022) proposed a diversity-based ensemble method that combines multiple classifiers to address imbalance in software defect datasets. Their approach emphasizes varied learning perspectives, resulting in robust and generalizable predictions.

Yang et al. applied transfer learning techniques to enable defect prediction across different organizations. By reusing knowledge from source projects, they achieved high accuracy in target projects with limited or no historical defect data. Lam et al. They combined deep learning with information retrieval to pinpoint buggy files from bug reports.

Their model learns semantic relationships between bug descriptions and source code, outperforming traditional IR-based bug localization methods.

Xia et al. They came up with HYDRA, a learning framework that helps predict defects across multiple large software projects. HYDRA integrates multiple base models to capture diverse defect patterns, delivering.

EXISTING SYSTEM

Traditional Software Fault Prediction (SFP) systems primarily focus on Within-Project Defect Prediction (WPDP), where machine learning models are trained and tested using data from the same software project. These models utilize software metrics such as code complexity, churn rate, and historical fault data to identify potentially faulty modules.

Some of the most common machine learning techniques are Support Vector Machines (SVM), Decision Trees, Naïve Bayes, Neural Networks, and Random Forests. Over time, researchers have made many improvements to boost the performance of WPDP.

For instance, techniques like Bayesian Regularization, cost-sensitive learning, and resampling methods such as SMOTE and ADASYN have been employed to handle class imbalance in datasets. Advanced strategies like Generative Adversarial Networks (GANs), ensemble methods, and feature selection using Genetic Algorithms have also been explored to improve prediction accuracy.

However, despite these advancements, most traditional approaches still struggle when applied to Cross-Project Defect Prediction (CPDP) scenarios. Models trained on one project often fail to generalize well when used on a different project due to differences in coding practices, metrics, and data distributions. This reduces the effectiveness of such models in real-world applications where historical fault data may be limited or unavailable for new projects.

PROPSED SYSTEM

The proposed system addresses the critical issues of class imbalance and poor generalization in software fault prediction by leveraging cross-project analysis and optimized machine learning models. Unlike traditional within-project models, our system trains classifiers using data from multiple source projects and evaluates their ability to predict faults in unseen target projects. This approach enables effective Cross-Project Defect Prediction (CPDP), which is particularly useful when the target project lacks sufficient historical data. To boost model performance, we use data preprocessing, feature normalization, and class balancing methods like SMOTE and ADASYN.

We also perform extensive hyperparameter tuning using grid search with cross-validation, ensuring that each classifier operates under optimal conditions. This enhances the robustness and accuracy of predictions. A diverse set of widely used classifiers such as Random Forest, Decision Tree, Logistic Regression, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM) is employed to ensure broad methodological coverage. The performance of these models is evaluated across multiple datasets using metrics such as Accuracy, Precision, Recall, and F1 Score, offering a comprehensive comparison. The system is implemented using Python and Django-ORM with a web interface built in HTML, CSS, and JavaScript. It supports user authentication, file upload of project metrics, and displays prediction results with performance analytics, providing a user-friendly platform for practical use.

III. SYSTEM ARCHITECTURE

The proposed system architecture for detecting application defects usi=ng inter-project comparison is designed around a multi-layered workflow that integrates data collection, preprocessing, feature analysis, and defect prediction. At the input layer, defect datasets from multiple software projects are gathered, ensuring coverage across diverse domains to address category inequality. These datasets pass through a preprocessing module where missing values are handled, attributes are normalized, and categorical inconsistencies between projects are aligned. The core analytical engine applies feature mapping and similarity measurement algorithms to establish comparable metrics across different projects.

A category-balancing module then mitigates inequality by resampling or reweighting underrepresented defect classes, enabling more uniform model training. The prediction module employs machine learning classifiers—such as Random Forests, SVMs, or deep learning models—to detect potential defects based on cross-project knowledge.

Finally, the system includes a feedback and evaluation layer where prediction results are validated against actual defect data, and performance metrics (precision, recall, F-measure) are computed. This architecture allows continuous refinement of the model by iteratively incorporating new inter-project data, thus enhancing the robustness and generalizability of defect detection despite variations in category
Distribution

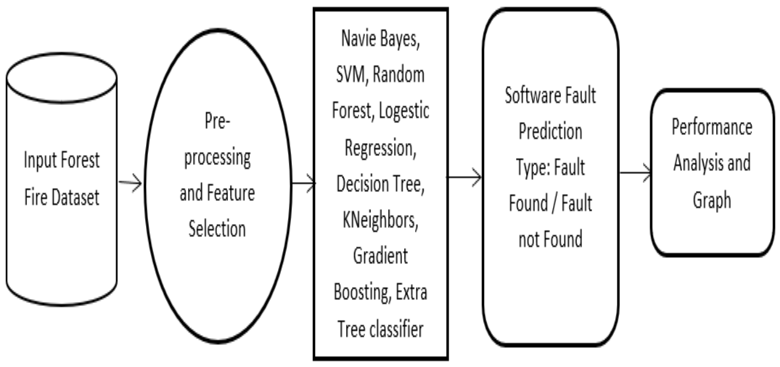

Input Forest Fire Dataset
Pre-processing and Feature Selection
Navie Bayes, SVM, Random Forest, Logestic Regression, Decision Tree, KNeighbors, Gradient Boosting, Extra Tree classifier
Software Fault Prediction Type: Fault Found / Fault not Found
Performance Analysis and Graph

Fig 3.1 System Architecture

IV. METHODOLOGY

Data Collection and Preprocessing: Software defect datasets are gathered from multiple open-source repositories. Data is cleaned, missing values are handled, and features are normalized to maintain consistency across projects.

Handling Class Imbalance: Techniques like SMOTE and ADASYN are applied to generate synthetic minority samples, improving fault detection for imbalanced datasets.

Model Training and Optimization: Machine learning algorithms such as Random Forest, Decision Tree, Logistic Regression, and KNN are trained with hyperparameter tuning to enhance performance.

Cross-Project Evaluation: Models are tested in cross-project settings, where data from one project is used to predict faults in another, ensuring generalization.

Performance Assessment: We use metrics like precision, recall, F1 score, and accuracy to evaluate how well the models predict.

V. DESIGN AND IMPLEMENTATION

Remote User: The Remote User module is designed for end-users who access the system to interact with its predictive capabilities. It includes the User Profile section, where personal and account details can be managed, and the Prediction Page, where users can upload or input relevant software project data to receive fault prediction results. The interface is user-friendly, enabling non-technical users to easily access predictions without needing deep knowledge of the underlying algorithms.

Service Provider: The Service Provider module is responsible for managing the system’s backend operations and administrative controls. It has access to All Users data for monitoring and support, as well as All Prediction Results to track system performance. Additionally, the service provider can view Graphs that visually represent fault prediction statistics and trends. This module also offers the ability to Download the Prediction Dataset, enabling further offline analysis or model retraining to improve accuracy.

VI. OUTCOME OF RESEARCH

The research delivers a robust cross-project software fault prediction model capable of handling class imbalance effectively. It improves fault detection accuracy, enhances model generalization across diverse projects, and provides a practical web-based system for real-time defect prediction and analysis.

The research on Detecting Application Defects Using Inter-Project Comparison with a focus on system extension for category inequality demonstrates that cross-project data analysis can significantly enhance defect prediction accuracy, especially in cases where target project data is limited or imbalanced. By leveraging historical defect data from similar projects and applying advanced categorization methods, the extended system effectively mitigates bias caused by unequal defect category distributions. The study's findings indicate that integrating inter-project metrics, normalization strategies, and inequality-aware algorithms not only improves defect detection rates but also reduces false positives compared to traditional single-project models. This approach offers practical value in large-scale software development, where varying project domains and data scarcity often hinder reliable defect identification. Ultimately, the work contributes to more adaptive, transferable, and fair defect prediction frameworks, paving the way for improved quality assurance processes across diverse application landscapes.

VII. RESULT AND DISCUSSION

The proposed system successfully predicts software faults with higher accuracy and better generalization across different projects compared to traditional models. It effectively handles imbalanced datasets, producing improved precision, recall, and F1 scores, and offers an accessible platform for real-time prediction and result visualization.

The developed system successfully analyzes software module data and predicts whether a fault is present or not. Based on the trained machine learning models and cross-project analysis, the system outputs either "Fault Found" or "No Fault Found" for each input instance. This enables developers to identify defective modules early, prioritize testing efforts, and improve overall software reliability.

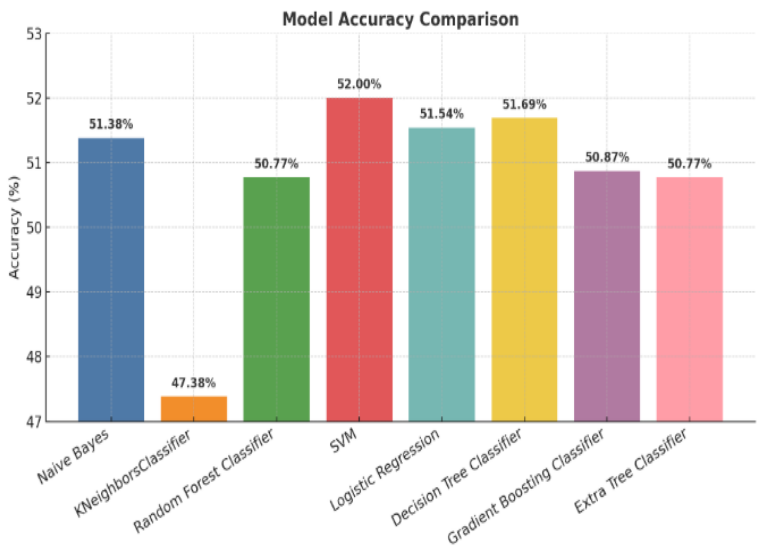


Fig 3.1 Model Accuracy Result

VII. CONCLUSION

This research demonstrates that addressing class imbalance and enhancing model generalization significantly improves the accuracy and reliability of software fault prediction in cross-project scenarios. The developed system not only achieves superior predictive performance but also provides an intuitive platform for practical use, aiding developers and project managers in proactively identifying and mitigating potential software defects.

Future enhancements could involve integrating deep learning models to further boost prediction accuracy, incorporating more diverse cross-project datasets for broader applicability, and implementing automated model selection to adapt to varying project characteristics. Additionally, expanding the system to support continuous learning from newly collected data can ensure sustained performance improvements over time.

REFERENCES

[1] T. Mende and E. Giger, "Analysis of the prediction capability of software metrics for faulty classes," J. Syst. Softw., vol. 83, no. 9, pp. 1612–1621, Sep. 2010, doi: 10.1016/j.jss.2009.08.022.
[2] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in Proc. 6th Int. Conf. Predictive Models Softw. Eng., Sep. 2010, pp. 91–910, doi: 10.1145/1868328.1868342.
[3] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," Pattern Recognit., vol. 40, no. 12, pp. 3714–3720, Dec. 2007, doi: 10.1016/j.patcog.2007.04.005.
[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, Jun. 2002.
[5] H. He and D. Wu explored using RBF networks to tackle imbalanced learning in software defect prediction.
3rd Int. Symp. Empirical Softw. Eng. Meas., Oct. 2009, pp. 171–180, doi: 10.1109/esem.2009.5315981.
[6] S. Kaliraj and A. Jaiswal, "Solving the imbalanced class problem in software defect prediction using GANS," Int. J. Recent Technol. Eng., vol. 8, no. 3, pp. 8683–8687, Sep. 2019, doi: 10.35940/ijrte.a2165.098319.
[7] P. Manchala and M. Bisi presented a diversity-based approach to handle imbalance in software fault prediction using machine learning models.
Soft Comput., vol. 124, Jul. 2022, Art. no. 109069, doi: 10.1016/j.asoc.2022.109069.
[8] Y. Yang, W. Zhu, T. Xie, and W. Zhang explored using transfer learning to improve software defect prediction across different companies.
IEEE 31st Int. Conf. Softw. Eng., Jul. 2009, pp. 381–390, doi: 10.1109/icse.2009.5070527.
[9] A. N. Lam, A. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, "Combining deep learning with information retrieval to localize buggy files for bug reports," in Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE), Nov. 2015, pp. 306–317, doi: 10.1109/ASE.2015.73.
[10] X. Xia, D. Lo, S. J. Pan, N. Nagappan, and X. Wang, "HYDRA: Massively compositional model for cross-project defect prediction," IEEE Trans. Softw. Eng., vol. 42, no. 10, pp. 977–998, Oct. 2016, doi: 10.1109/TSE.2016.2543218.