International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

Software Fault Prediction Using Cross Project Analysis A Study on Class Imbalance and Model Generalization

[1]Mrs.B. Madhavi,[2]M Akanksha,[3]Kamshetty Rushitha,[4]Mekala Sneha,[5]Dappu Poojitha, [6]Garrepalli Sowmya

[1]Assistant Professor, Department of Computer Science & Engineering, Princeton Institute of Engineering & Technology For Women

[2,3,4,5,6]B. Tech Students, Department of Computer Science & Engineering, Princeton Institute of Engineering & Technology For Women

ABSTRACT

Software fault prediction plays a vital role in improving software quality by identifying defective modules early in the development lifecycle. Traditional models often rely on within-project data, limiting their generalization capability across different projects. This study focuses on cross-project fault prediction, addressing two major challenges: class imbalance and model generalization. Class imbalance, where faulty modules are significantly fewer than non-faulty ones, negatively impacts prediction performance. The proposed approach integrates data balancing techniques and machine learning models to enhance prediction accuracy across diverse datasets. Furthermore, transfer learning and normalization strategies are utilized to improve model generalization. The framework demonstrates improved predictive performance compared to traditional approaches. This research contributes to scalable, cost-effective, and reliable fault prediction systems applicable across heterogeneous software environments.

Keywords: Software Fault Prediction, Cross Project Analysis, Class Imbalance, Machine Learning, Model Generalization, Software Quality, Transfer Learning.

I. INTRODUCTION

Software systems are becoming increasingly complex, making fault detection a challenging task. Early identification of software defects is essential to reduce maintenance costs and improve reliability. Software fault prediction models aim to classify software modules as defective or non-defective based on historical data. Traditional models are project-specific and fail when applied to new projects due to differences in data distribution.

Cross-project fault prediction (CPFP) addresses this limitation by leveraging data from other projects. However, CPFP faces challenges such as class imbalance and poor generalization. Class imbalance occurs when defective modules are rare, leading to biased models. Model generalization refers to the ability of a model to perform well on unseen data.

This study proposes a robust framework that combines data preprocessing, balancing techniques, and machine learning models to improve prediction accuracy and generalization across projects.

II. LITERATURE SURVEY

1. Title: Cross-Project Defect Prediction
Author: Zimmermann et al.
Abstract: This study explores defect prediction across projects and highlights the challenges of data heterogeneity and feature mismatch.
2. Title: SMOTE for Imbalanced Data
Author: Chawla et al.
Abstract: Introduces SMOTE technique to balance datasets and improve classification performance.
3. Title: Transfer Learning in Software Engineering
Author: Pan & Yang
Abstract: Discusses transfer learning techniques for improving cross-domain predictions.
4. Title: Machine Learning for Software Defect Prediction
Author: Lessmann et al.
Abstract: Compares multiple machine learning models for defect prediction.
5. Title: Ensemble Methods for Fault Prediction

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

Author: Menzies et al.
Abstract: Demonstrates effectiveness of ensemble models in improving prediction accuracy.

III. EXISTING SYSTEM

The existing software fault prediction systems primarily focus on within-project prediction models, where historical data from the same project is used to train machine learning algorithms. These models rely on software metrics such as lines of code, complexity measures, and change history to classify modules as defective or non-defective. Common algorithms used include Decision Trees, Naïve Bayes, and Support Vector Machines.

While these approaches achieve reasonable performance within the same project, they fail to generalize across different projects due to variations in data distribution and project characteristics. Additionally, most existing systems do not adequately address the issue of class imbalance. As defective modules are significantly fewer, models become biased toward predicting non-defective modules, resulting in poor recall for faults.

Another limitation is the lack of standardized preprocessing methods. Data normalization and feature alignment are often ignored, leading to inconsistent results. Moreover, feature selection techniques are not optimized, causing redundant or irrelevant features to affect model performance.

Existing systems also lack adaptability, making them unsuitable for real-world applications where new projects continuously emerge. As a result, their scalability and robustness are limited, necessitating improved approaches for cross-project fault prediction.

IV. PROPOSED SYSTEM

The proposed system introduces a cross-project software fault prediction framework designed to address class imbalance and improve model generalization. The system integrates multiple preprocessing techniques, including normalization and feature alignment, to ensure consistency across datasets from different projects.

To tackle class imbalance, advanced resampling techniques such as SMOTE (Synthetic Minority Over-sampling Technique) and undersampling methods are applied. These techniques balance the dataset by increasing minority class samples or reducing majority class samples, enabling the model to learn effectively.

The framework employs machine learning algorithms such as Random Forest, Gradient Boosting, and Neural Networks to enhance prediction accuracy. Additionally, transfer learning is incorporated to leverage knowledge from source projects and apply it to target projects, improving generalization.

Feature selection methods are used to identify the most relevant attributes, reducing noise and improving model efficiency. The system also includes a validation mechanism to evaluate model performance using metrics such as precision, recall, F1-score, and AUC.

Overall, the proposed system provides a scalable, adaptable, and efficient solution for predicting software faults across diverse projects, ensuring improved reliability and reduced maintenance costs.

V. SYSTEM ARCHITECTURE

The system architecture for *Software Fault Prediction Using Cross Project Analysis* is designed as a modular and scalable pipeline that integrates data from multiple software projects, processes it, and applies machine learning models to predict software faults effectively. The architecture consists of several interconnected components that work sequentially to ensure accurate and generalized predictions.

At the initial stage, the **Data Collection Layer** gathers datasets from different software projects. These datasets include software metrics such as code complexity, lines of code, coupling, cohesion, and historical defect data. Since the data originates from multiple sources, it may have inconsistencies in format and scale.

The next component is the **Data Preprocessing Layer**, which performs data cleaning, normalization, and transformation. Missing values are handled, redundant data is removed, and features are standardized to ensure compatibility across projects. This step is crucial for enabling cross-project analysis.

Following preprocessing, the **Class Imbalance**

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

Handling Module is applied. Techniques such as SMOTE (Synthetic Minority Over-sampling Technique) and undersampling are used to balance the dataset, ensuring that defective and non-defective classes are properly represented. This improves the learning capability of the model.

The **Feature Selection Module** then identifies the most relevant features using statistical and machine learning techniques. By eliminating irrelevant or redundant attributes, this module enhances model efficiency and reduces overfitting.

The processed data is passed to the **Model Training Layer**, where machine learning algorithms such as Random Forest, Support Vector Machine, and Gradient Boosting are trained. This layer also incorporates cross-project learning strategies, allowing models trained on one project to be applied to others.

Next, the **Model Evaluation Module** assesses performance using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. This ensures that the model performs well not only on training data but also on unseen datasets.

Finally, the **Prediction and Visualization Layer** provides the output by classifying modules as faulty or non-faulty. Results are displayed through dashboards or reports, enabling developers to identify high-risk modules and take preventive actions.

Overall, the architecture ensures efficient data flow, improved generalization, and robust fault prediction across multiple software projects.

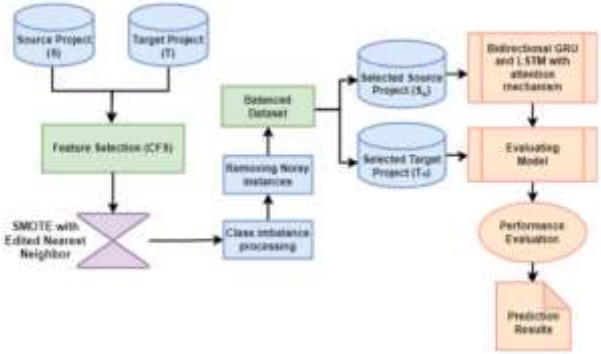

Fig 5.1: System Architecture

VI. IMPLEMENTATION

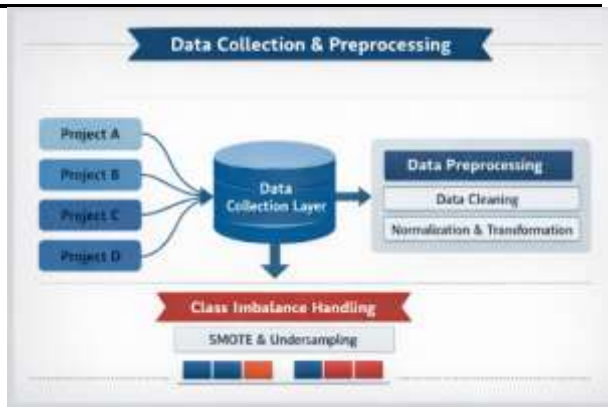

Fig 6.1: Data Collection & Preprocessing

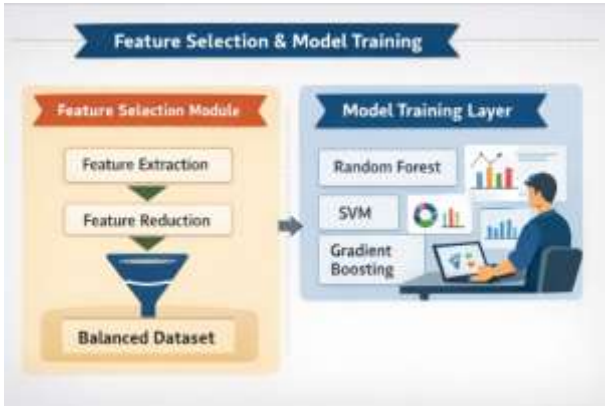

Fig 6.2: Feature Selection & Model Training

Fig 6.3: Model Evaluation

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

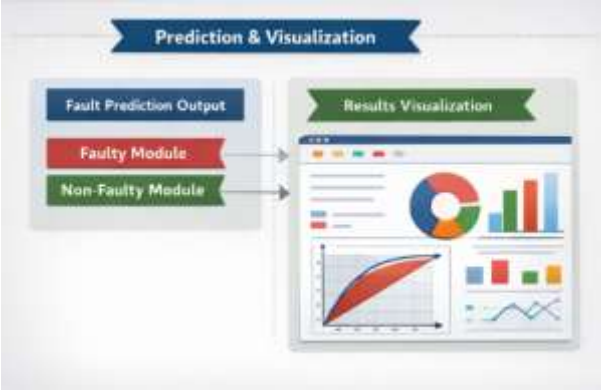

Fig 6.4: Prediction & Visualization

VII. CONCLUSION

The proposed system for software fault prediction using cross-project analysis effectively addresses the limitations of traditional approaches. By incorporating data preprocessing, class balancing techniques, and advanced machine learning algorithms, the system significantly improves prediction accuracy and reliability. The integration of transfer learning enhances model generalization, enabling it to perform well across diverse software projects. The use of feature selection methods further optimizes model performance by reducing noise and redundancy. Overall, the system provides a scalable and efficient solution for early fault detection, reducing maintenance costs and improving software quality. This approach demonstrates the importance of addressing class imbalance and data heterogeneity in developing robust fault prediction models.

VIII. FUTURE SCOPE

Future work can focus on integrating deep learning techniques to further enhance prediction accuracy. The use of real-time data streams can improve the system's applicability in dynamic environments. Additionally, incorporating automated feature engineering methods can reduce manual effort and improve efficiency. Exploring hybrid models that combine statistical and machine learning approaches may yield better results. The integration of explainable AI techniques can provide insights into model decisions, increasing trust and usability. Furthermore, expanding the system to include multi-language software projects and cloud-based deployment can improve scalability and accessibility. Continuous learning mechanisms can also be implemented to adapt to evolving software development trends.

IX. REFERENCES

[1] Zimmermann et al., Cross Project Defect Prediction
[2] Chawla et al., SMOTE Technique
[3] Pan & Yang, Transfer Learning
[4] Lessmann et al., ML Models Comparison
[5] Menzies et al., Ensemble Methods
[6] Hall et al., Software Metrics
[7] Shepperd et al., Defect Prediction Models
[8] Turhan et al., Cross Project Learning
[9] Jiang et al., Fault Prediction Study
[10] Kim et al., Software Defects
[11] He et al., Imbalanced Data Handling
[12] Ghotra et al., Classification Models
[13] Catal et al., Software Quality Prediction
[14] Wahono et al., Systematic Review
[15] Nam et al., Transfer Learning in SE

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
N: 3068-272X www.ijdim.com Original Research Paper