ISSN (Online): 3139-7689

Available online: www.ijsrmes.com

International Journal for Scientific Research in Modern Engineering and Science

Published in Conference Special Issue: Vol. 1, Special Issue 1, pp. 271–274, 2026

SOFTWARE BUG PREDICTION USING SUPERVISED MACHINE LEARNING ALGORITHMS

Sujal Raj[1*], Priyal Chouhan[2], Bhumi Sahayata[3], Suryamani Singhaniya[4]

[1234]*Student, Computer Science and Engineering, Shri Shankaracharya Technical Campus, Bhilai*

Abstract

Software systems are becoming increasingly large and complex due to rapid technological growth and continuous feature expansion. As complexity increases, the probability of defects or bugs in the software also increases. Software bugs reduce reliability, increase maintenance cost, and may cause system failures after deployment. Therefore, predicting defects during early development stages has become an important research problem in software engineering.

This research focuses on predicting software bugs using supervised machine learning algorithms. Four classification techniques are implemented: Logistic Regression, Naïve Bayes, Decision Tree, and Random Forest. Historical software project data containing software metrics such as Lines of Code, Cyclomatic Complexity, Coupling, and Inheritance depth is used to train and test the models. The performance of each classifier is evaluated using Accuracy, Precision, Recall, F1-score, and 10-Fold Cross Validation.

Experimental results show that ensemble learning through Random Forest provides better prediction performance compared to individual classifiers. The study demonstrates that machine learning-based bug prediction models can assist development teams in identifying high-risk modules early, improving software quality, reducing testing effort, and minimizing development cost.

Keywords: *Software Bug Prediction, Supervised Learning, Random Forest, Defect Prediction, Software Quality*

1. Introduction

Software plays a critical role in modern society. From banking systems to healthcare applications and e-commerce platforms, reliable software is essential. However, developing completely error-free software is almost impossible. Bugs or defects are common during software development due to human mistakes, poor design, or complex logic. Traditional defect detection techniques mainly rely on manual testing, code reviews, and debugging processes. These methods are effective but time-consuming and expensive. In large-scale projects, it becomes difficult to test every module with equal effort. As a result, identifying which modules are more likely to contain defects becomes important. Software bug prediction aims to identify defect-prone modules before testing or deployment. By predicting risky components, development teams can allocate resources more efficiently and focus testing efforts where they are most needed.

Machine Learning (ML) has shown promising results in predictive analysis across many domains. In software engineering, ML models can learn patterns from historical data and predict whether a software module is

likely to be defective. This research explores supervised machine learning algorithms for software bug prediction and compares their performance.

The main objectives of this study are:

- To implement supervised learning models for defect prediction
- To identify the most effective model for defect prediction
- To compare Logistic Regression, Naïve Bayes, Decision Tree, and Random Forest
- To evaluate models using cross-validation

2. Literature Review

Several researchers have worked on software defect prediction over the years. Early studies focused on statistical models and software metrics. Menzies et al. (2007) showed that static code attributes can be used to predict software defects effectively. Fenton and Neil (1999) criticized some prediction models and emphasized the importance of proper validation. Later, ensemble methods such as Random Forest gained popularity because they reduced overfitting and improved accuracy. Many studies have concluded that ensemble techniques outperform single classifiers in defect prediction tasks. However, there is still a need to compare multiple classifiers on the same dataset using consistent evaluation metrics. This research contributes by providing a comparative study using standard validation techniques.

Dataset Description

The dataset used in this study contains historical software project data. Each record represents a software module with various software metrics.

Some important features include:

- Lines of Code (LOC)
- Cyclomatic Complexity
- Depth of Inheritance Tree (DIT)
- Coupling Between Objects (CBO)
- Number of Methods
- Number of Attributes

Each module is labelled as:

- → Non-defective
- → Defective

Data Preprocessing

Before training the models, preprocessing steps were applied:

1. Removal of missing values
2. Feature normalization
3. Data balancing (if required)

4. Train-Test split (80%-20%)

Data preprocessing is an important stage in any machine learning project because raw data collected from software repositories may contain missing values, inconsistent formats, or irrelevant information. Proper preprocessing improves the quality of the data and helps the models learn more effectively. Proper preprocessing significantly improves the reliability and performance of the prediction models. To solve this issue, balancing techniques such as oversampling or under sampling can be applied so that both classes are fairly represented during training.

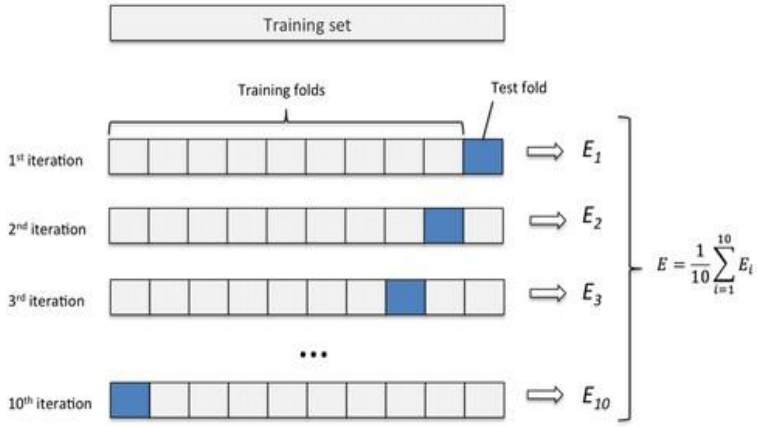

Figure 1. Reliability and performance of the prediction models

3. Results and Discussion

After training and evaluation, the following results were observed:

Algorithm	Accuracy	Precision	Recall	F1-Score
Logistic Regression	82%	90%	75%	77%
Naïve Bayes	78%	74%	72%	73%
Decision Tree	85%	83%	80%	81%
Random Forest	90%	88%	86%	87%

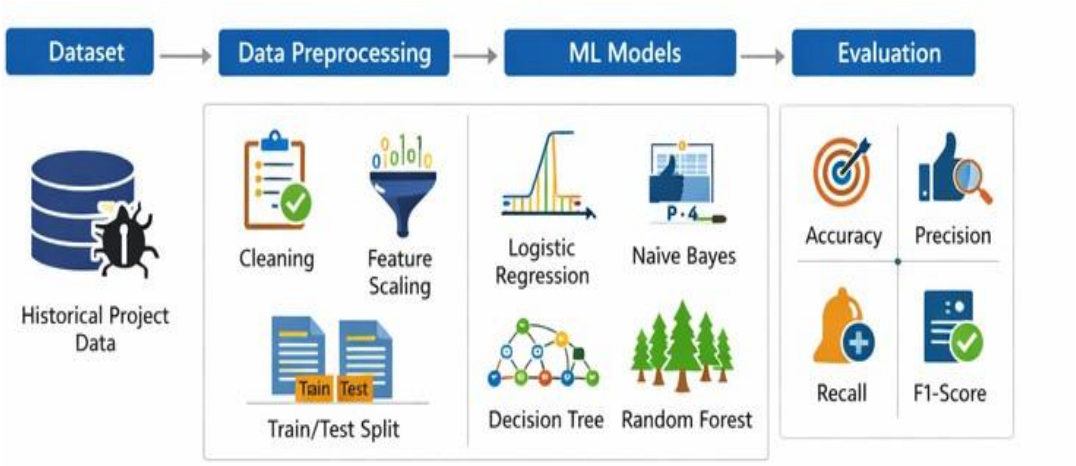

Figure 2. Software bug prediction framework

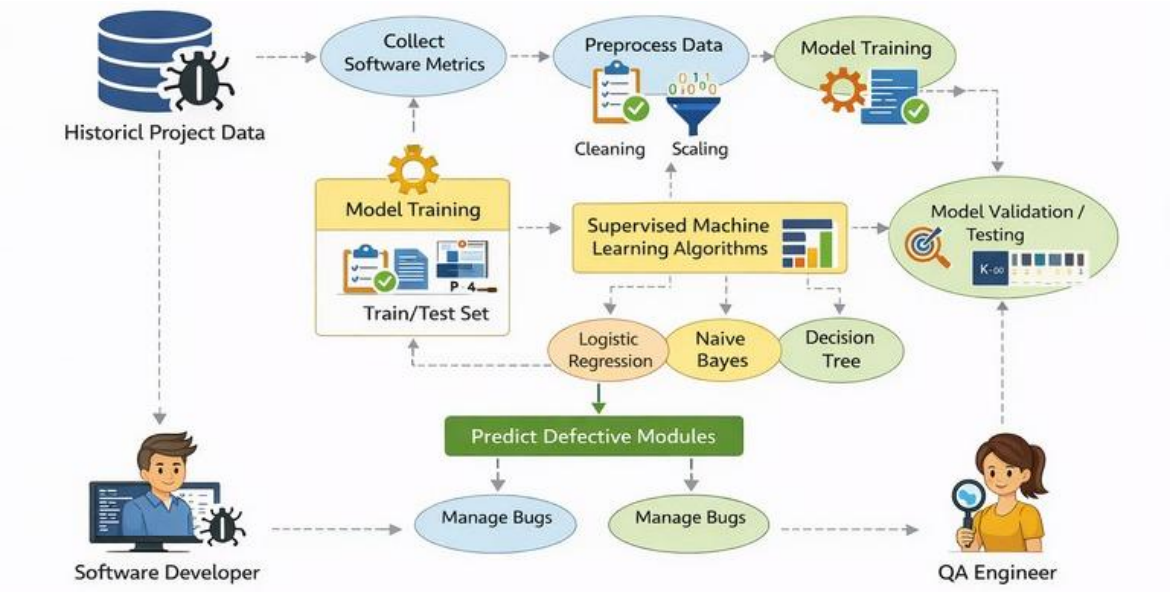

Figure 3. Software bug prediction using supervised machine learning

4. Conclusion

This research presented a comparative study of supervised machine learning algorithms for software bug prediction. Logistic Regression, Naïve Bayes, Decision Tree, and Random Forest were implemented and evaluated using cross-validation. Experimental results show that Random Forest provides the best performance in terms of accuracy and stability. The findings confirm that machine learning-based defect prediction can significantly improve software development processes. By predicting high-risk modules early, organizations can reduce maintenance cost, increase reliability, and deliver higher- quality software products.

References

[1] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," IEEE Transactions on Software Engineering, 2007.

[2] N. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," IEEE Transactions on Software Engineering, 1999.

[3] L. Breiman, "Random Forests," Machine Learning Journal, 2001.

[4] I. H. Witten, E. Frank, and M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2011.

[5] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, Elsevier, 2012.

[6] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A Systematic Literature Review on Fault Prediction Performance," IEEE Transactions on Software Engineering, 2012.

[7] Y. Zhang and X. Xie, "Software Fault Prediction Based on Machine Learning: A Survey," Journal of Systems and Software, 2019.