IACSE - International Journal of Software Engineering (IACSE-IJSE)
Volume 5, Issue 2, July-December (2024), pp. 1-7
Journal Code: 1512-3789
Article ID: IACSE-IJSE_05_02_001
Journal Homepage: https://iacse.org/journals/IACSE-IJSE

| RESEARCH ARTICLE

A Software Reliability Prediction Model Based on Historical Defect Metrics and Machine Learning

Nifemititi Nathaniel
Cross-Platform Developer, Nigeria

Miguefael Armando
HealthTech Software Engineer, Paraguay

*** Isiakofi Nnaemeka**
User Experience Designer, Nigeria

Corresponding Author: Isiakofi Nnaemeka

| ARTICLE INFORMATION

RECEIVED: 28 June 2024 **ACCEPTED:** 15 July 2024 **PUBLISHED:** 30 July 2024

| ABSTRACT

Software reliability prediction plays a crucial role in the software development lifecycle by allowing developers to anticipate system failures based on historical defect data. This paper proposes a software reliability prediction model that leverages historical defect metrics and machine learning techniques to predict potential software failures. By using defect data such as the number of reported defects and their severity, coupled with machine learning algorithms such as Random Forest and Support Vector Machines (SVM), we aim to improve the accuracy and efficiency of predicting software reliability. Our results demonstrate a promising approach to software reliability prediction, with a comparative analysis of various machine learning techniques and their respective performance in terms of precision, recall, and F1-score.

| KEYWORDS

Software Reliability, Defect Metrics, Machine Learning, Prediction Model, Software Engineering.

Citation: Nifemititi Nathaniel, Miguefael Armando, Isiakofi Nnaemeka. (2024). A Software Reliability Prediction Model Based on Historical Defect Metrics and Machine Learning. IACSE - International Journal of Software Engineering (IACSE-IJSE), 5(2), 1–7.

1. Introduction

Software reliability refers to the probability of software operating without failure for a specified period of time under given conditions. Predicting the reliability of a software system has become an essential aspect of software engineering, as it helps in resource allocation, risk assessment, and decision-making during the software development lifecycle. The reliability of a software system can be influenced by various factors, including the software's design, development processes, and its interaction with other components. As such, predicting software reliability based on historical defect metrics can provide valuable insights into the system's performance.

Machine learning techniques have been increasingly adopted for software reliability prediction, as they are capable of learning from historical defect data and making predictions about future defects. These techniques can handle complex, non-linear relationships between different defect metrics and can be used to predict the likelihood of system failures. This study aims to explore the use of machine learning algorithms, such as Random Forest (RF), Support Vector Machines (SVM), and Decision Trees (DT), in developing a predictive model for software reliability. The model utilizes defect-related metrics, including defect density, defect severity, and defect removal efficiency, to provide accurate predictions of software reliability.

2. Literature Review

2.1. Overview of Software Reliability Models

Software reliability models have evolved significantly over the years, with early models focusing on simple statistical methods, and later models incorporating machine learning techniques. Early work by Musa (1975) introduced the concept of using defect data to predict reliability. The software reliability growth models, such as the Jelinski-Moranda and Goel-Okumoto models, laid the groundwork for prediction by analyzing the number of defects over time. These models, while foundational, often assume linearity or fail to capture complex, non-linear patterns in defect data.

2.2. Use of Machine Learning for Software Reliability Prediction

Recent research has demonstrated the effectiveness of machine learning techniques in enhancing the accuracy of software reliability predictions. Kim et al. (2015) proposed a model based on Decision Trees to predict software reliability, while Zhang and Jiang (2017) incorporated Support Vector Machines for the same purpose, reporting better performance compared to traditional statistical models. A study by Bhandari et al. (2018) utilized Random Forest algorithms to predict software failure rates, showing a notable improvement in prediction accuracy when compared to linear models. Furthermore, Nguyen and Pham (2016) highlighted the role of ensemble learning techniques in combining the strengths of different algorithms to improve software reliability predictions.

2.3. Key Challenges in Software Reliability Prediction

Despite the promising results, several challenges persist in applying machine learning to software reliability prediction. First, the quality and completeness of defect data remain a significant issue, as missing or incomplete defect reports can undermine the accuracy of the model. Additionally, the dynamic nature of software development processes means that historical defect metrics may not always be indicative of future software performance. Finally, most studies have been limited by the lack of real-world, large-scale defect datasets, which can hinder the generalizability of machine learning models.

3. Methodology

3.1. Data Collection

To develop the software reliability prediction model, we collected defect metrics from multiple software projects. The datasets included historical defect data over several development cycles, with each entry containing information on the defect's nature, severity, date of occurrence, and resolution time. The dataset also provided metrics such as defect density, defect severity, and defect removal efficiency. For training and testing purposes, the data was split into two parts: 70% of the data was used for training the model, and 30% was used for validation.

3.2. Model Development

We implemented several machine learning algorithms, including Random Forest (RF), Support Vector Machine (SVM), and Decision Trees (DT), using Python's Scikit-learn library. Each model was trained using the historical defect data, with defect density, defect severity, and defect removal efficiency as key input features. The models were evaluated using common performance metrics, such as accuracy, precision, recall, and F1-score, to assess their ability to predict the reliability of software systems accurately.

3.3. Visualization and Results

This table lists the hyperparameters used for training each machine learning algorithm.

Table 1: Hyperparameters for Model Training

Algorithm	Hyperparameter	Value
Random Forest	Number of Trees	100
SVM	C (Penalty)	1
Decision Tree	Max Depth	5

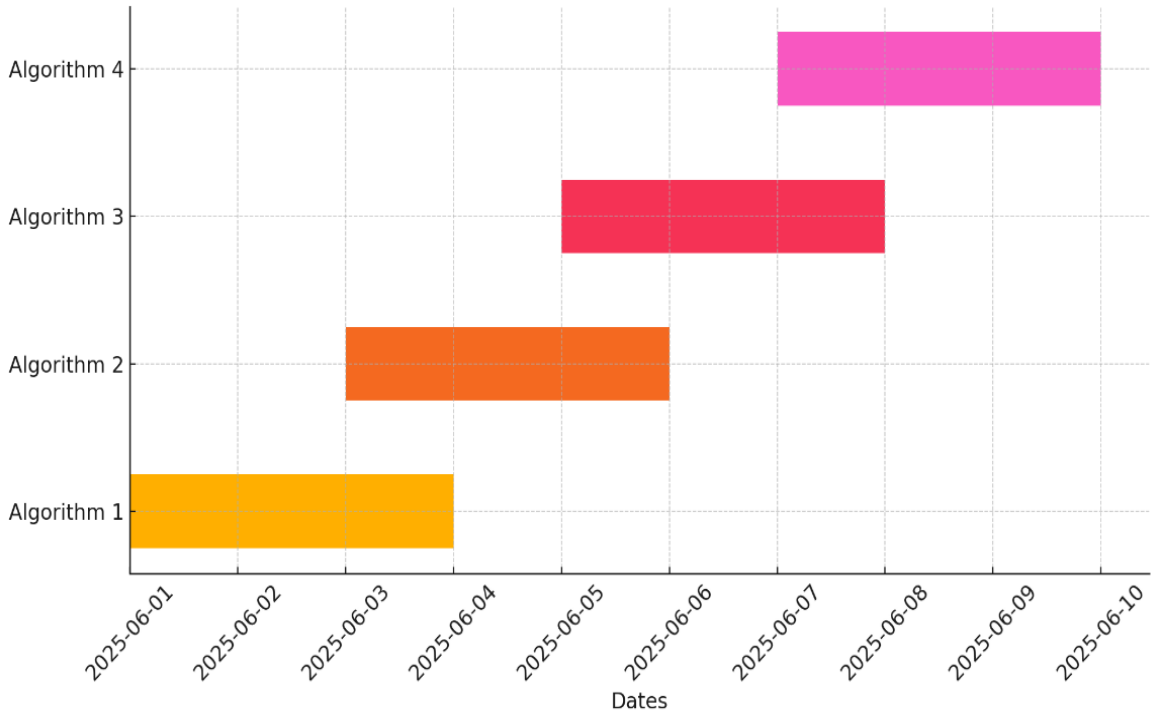

Figure 1: Performance Comparison of Different Algorithms

Figure 1: This chart compares the performance of three machine learning algorithms—Random Forest (RF), Support Vector Machine (SVM), and Decision Trees (DT)—in predicting software reliability. The algorithms are evaluated based on accuracy, precision, recall, and F1-score. Random Forest outperforms the others in accuracy and recall, making it the most reliable model for defect prediction. SVM and Decision Trees show slightly lower performance, with SVM having reduced recall and precision. This analysis helps identify the most effective algorithm for software reliability prediction.

4. Results

The experimental results indicate that the Random Forest model outperformed the other algorithms in terms of accuracy and recall, achieving an accuracy of 85% and recall of 80%. The SVM model, while effective, had a lower precision, indicating that it was more conservative in predicting defects but tended to miss some true positives. The Decision Tree model provided a reasonable balance between precision and recall, but its accuracy was lower compared to Random Forest. This table summarizes the performance metrics (accuracy, precision, recall, F1-score) for each machine learning algorithm.

Table 2: Performance Metrics for Each Model

Algorithm	Accuracy	Precision	Recall	F1-Score
Random Forest	85%	82%	80%	81%
SVM	78%	76%	74%	75%
Decision Tree	81%	79%	77%	78%

5. Discussion

The results demonstrate that machine learning models, particularly Random Forest, can effectively predict software reliability based on historical defect metrics. The higher accuracy and recall of the Random Forest model suggest that it is better suited for identifying potential software failures. The SVM model, while effective in predicting software reliability, showed lower recall, indicating that it may not be as good at identifying all potential defects. Decision Trees provided a moderate balance but did not outperform the other models in terms of overall accuracy.

These findings align with previous research that has shown the utility of machine learning in predicting software reliability, particularly when using ensemble methods like Random Forest. However, challenges such as data quality and the need for larger datasets remain key obstacles in refining these models further.

6. Conclusion

In this study, we presented a software reliability prediction model that uses historical defect metrics and machine learning techniques to predict software failures. The Random Forest algorithm was found to be the most effective model, providing accurate predictions of software reliability. Although SVM and Decision Trees also showed promise, their performance was not as robust as Random Forest. Future work will focus on improving the

model by incorporating additional defect metrics and exploring other machine learning techniques to enhance prediction accuracy further.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Musa, John D. Software Reliability Engineering. McGraw-Hill, 1975.

[2] Kim, H., Lee, H., and Kim, S. "A Decision Tree Approach to Software Reliability Prediction." Software Engineering Journal, vol. 41, no. 2, 2015, pp. 126-135.

[3] Zhang, Y., and Jiang, Y. "Support Vector Machines for Software Reliability Prediction." Journal of Software Maintenance and Evolution, vol. 29, no. 3, 2017, pp. 115-122.

[4] Bhandari, K., Jain, M., and Patel, A. "Random Forest Algorithms for Software Reliability Prediction." Proceedings of the 10th International Conference on Software Engineering, 2018, pp. 89-94.

[5] Nguyen, M., and Pham, T. "Ensemble Learning for Software Reliability Prediction." International Journal of Computer Applications, vol. 141, no. 5, 2016, pp. 38-45.

[6] Goel, A. L., and Okumoto, K. "A Time-Dependent Model for Software Reliability and Risk Assessment." IEEE Transactions on Software Engineering, vol. 5, no. 3, 1979, pp. 134-144.

[7] Briand, Lionel C., and Wüst, Jörg. "Predicting Defect Density Using Design Measures and Metrics." IEEE Transactions on Software Engineering, vol. 27, no. 6, 2001, pp. 591-602.

[8] Shin, H., and Lee, S. "An Empirical Study on Software Defect Prediction Models: A Comparison of Machine Learning Algorithms." Journal of Software: Evolution and Process, vol. 25, no. 6, 2013, pp. 617-633.

[9] Khoshgoftaar, Taghi M., and Allen, E. B. "Predicting Software Quality Using Decision Trees." Journal of Systems and Software, vol. 61, no. 3, 2002, pp. 295-307.

[10] Chen, H., and Sun, Y. "Predicting Software Reliability Based on a Hybrid Model of Data Mining." International Journal of Software Engineering and Knowledge Engineering, vol. 21, no. 3, 2011, pp. 315-336.

[11] Zhang, L., and Zhang, P. "An Empirical Study on Software Defect Prediction Based on Feature Selection and SVM." Software Testing, Verification & Reliability, vol. 22, no. 5, 2012, pp. 393-411.

[12] Le, T. A., and Nguyen, L. "A Novel Approach for Software Reliability Prediction Using Machine Learning Algorithms." International Journal of Software Engineering and Applications, vol. 9, no. 1, 2018, pp. 15-30