International Journal of Sustainable Development Through AI, ML and IoT
Volume 3 | Issue2 | 2024
https://ijsdai.com/index.php/IJSDAI/index
ISSN (Online): 2584-0827

Leveraging Clustering Algorithms for Optimizing Test Case Prioritization in Software Development

[1]Sheetal Sharma, [2] Swati V. Chande and [3]N.K. Joshi
[1]Research Scholar, Rajasthan Technical University, Kota-324010, India
[2]International School of Informatics & Management, Jaipur 302020, India
[3]Modi Institute of Management & Technology, Kota- 324009, India
sheetljoshi2e@gmail.com
* Corresponding author

ARTICLE INFO
Received: 13 Sep 2024
Revised: 02 Oct 2024
Accepted: 05 Oct 2024

ABSTRACT

In the field of software development, ensuring the quality and reliability of software through rigorous testing is crucial. However, as software systems grow more complex, managing the vast number of test cases becomes increasingly challenging. To address this issue, test case prioritization techniques are essential, as they help identify and execute the most critical test cases first. This research paper proposes a novel approach to test case prioritization by leveraging clustering techniques, specifically K-means, in conjunction with machine learning algorithms. We investigate the advantages of K-means clustering for grouping similar test cases and enhancing prioritization efficiency. Additionally, we explore various machine learning algorithms, including Decision Trees (DT), Random Forests (RF), and Neural Networks (NN), and compare their effectiveness against traditional prioritization methods such as code coverage-based and risk-based techniques. Our study evaluates these methods using multiple datasets and metrics, including the number of test cases executed, fault detection rate, and execution time. The experimental results reveal that integrating K-means clustering with machine learning algorithms can significantly improve test case prioritization by reducing the number of test cases executed while maintaining or even enhancing fault detection rates. We also discuss the limitations of our approach and suggest directions for future research in optimizing test case prioritization with advanced machine learning techniques. Our findings offer valuable insights into developing more efficient software testing practices, ultimately contributing to the enhanced quality and reliability of software systems.

.

Introduction

Software testing plays a crucial role in software development by verifying the quality and dependability of software systems. The process includes running multiple test cases to uncover defects and issues within the software. As software systems grow in complexity, the number of required test cases also rises, making it difficult to run all tests within constrained time and resource limits. To manage this challenge, test case prioritization techniques have been developed to focus on executing the most vital test cases first.

Traditional methods for prioritizing test cases often rely on criteria such as code coverage or risk assessment. While these approaches have their merits, they also come with limitations. For instance, prioritizing based on code coverage may not always highlight the most crucial test cases, and risk-based prioritization can be influenced by the subjective assessments of testers, potentially leading to incomplete coverage.

We propose leveraging machine learning (ML) algorithms for enhancing test case prioritization. Our research focuses on evaluating the effectiveness of various ML algorithms, including decision trees, random forests, and neural networks, and comparing their performance to traditional prioritization methods. Additionally, we suggest that integrating ML algorithms for predicting priorities, followed by applying K-means clustering, could lead to improved Average Percentage of Faults Detected (APFD) rates. This study aims to determine the most effective ML approach for test case prioritization, identify its limitations, and explore future research opportunities. Our findings are intended to advance software testing techniques, thereby enhancing the quality and reliability of software systems.

Prioritizing test cases plays a vital role in software testing, especially for complex software systems. Conventional techniques, such as code coverage, risk analysis, and fault-proneness, are commonly used for test case prioritization. While these methods have proven beneficial, they often encounter challenges like subjectivity and limited coverage.

In recent years, machine learning algorithms have gained attention as a promising solution for improving the efficiency and accuracy of test case prioritization. By leveraging data analysis, these algorithms can detect patterns and provide insightful predictions, leading to more informed decision-making. Several studies have investigated the potential of machine learning to enhance test case prioritization practices.

Literature Review

Test case prioritization plays a vital role in software testing, particularly for large and complex software systems. Various approaches have been developed to prioritize test cases using factors like code coverage, risk, and fault-proneness. While these methods are valuable, they often come with challenges, such as subjective decision-making and incomplete coverage.

Recently, machine learning algorithms have emerged as a promising approach to enhance the efficiency and accuracy of test case prioritization. These algorithms can autonomously analyze data, identify patterns, and make predictions or informed decisions. Numerous studies have explored the application of machine learning techniques to improve the process of test case prioritization.

Abid R and Nadeem A [1] proposed a multi-criteria approach for test case prioritization, which improved performance by accounting for factors like fault detection and code coverage. This approach demonstrated superior results compared to traditional prioritization methods.

Khatibsyarbini M, et al. [2] conducted a systematic review of test case prioritization methods in regression testing, concluding that modern techniques, particularly those using machine learning, are more effective than conventional strategies for handling large test suites and enhancing fault detection.

Ammar A, et al. [3] introduced an enhanced weighted method for test case prioritization that uses unique priority values, which optimized the efficiency of regression testing and outperformed traditional methods by improving execution time and fault detection.

Konsaard P and Ramingwong L [4] applied the artificial bee colony algorithm to prioritize test cases based on code coverage. This optimization-based approach resulted in better fault detection and test execution efficiency compared to standard methods.

Rosero RH, et al. [5] focused on the regression testing of database applications in an incremental development setting. Their method improved fault detection and reduced testing time, making it more efficient than traditional regression testing techniques.

Zhu Q, et al. [6] explored k-means clustering for test case prioritization, which grouped similar test cases to optimize the test suite. This technique led to better fault detection and code coverage, demonstrating the potential of clustering for improving test case prioritization.

Wang S, et al. [7] utilized fuzzy logic to prioritize test cases, considering factors like risk, code complexity, and execution time. Their adaptive approach outperformed fixed traditional methods by offering greater flexibility and improved performance in fault detection.

Srikanth H, et al. [8] proposed a machine learning-based method using decision trees for prioritizing test cases. Their model leveraged historical data to adjust prioritization dynamically, achieving higher fault detection rates than traditional heuristic methods.

Jiang H, et al. [9] developed a hybrid approach combining fuzzy logic and machine learning techniques like neural networks. This combination allowed for more precise and adaptive test case prioritization, significantly improving performance compared to traditional approaches.

Zheng Y, et al. [10] integrated k-means clustering with fuzzy logic for test case prioritization, grouping test cases by similarity and assigning priority using fuzzy logic. This hybrid model enhanced fault detection and reduced test execution time, outperforming conventional methods.

Proposed Methodology

The proposed framework for Test Case Prioritization (TCP) seeks to tackle the complexities and subjectivity involved in prioritizing test cases within the software testing process. Effective prioritization is crucial in software development as it helps developers and testers allocate resources more efficiently, reduce costs, and speed up the testing process . TCP also plays a key role in improving fault detection in regression testing by ensuring that critical test cases are executed earlier in the testing cycle. Our framework uses a multi-dimensional approach to prioritize test cases based on a combination of key factors .

In this approach, priority will be assigned by focusing on essential criteria such as execution time, the number of detected faults, and the severity of those faults. While prior studies have considered factors like defect history and time, or severity and time, these approaches often have limitations. Our framework aims to overcome these shortcomings by incorporating the most relevant factors for a more robust prioritization process.

These factors include historical bug data, code complexity, time constraints, and potentially other context-specific elements. Fuzzy logic plays a crucial role in the framework, enabling a flexible and intuitive way to manage imprecise or uncertain data, which is commonly encountered in software testing .

Once test cases are prioritized using the fuzzy logic classifier, the next step involves applying clustering algorithms to further refine the prioritization. Techniques such as K-Means clustering or other clustering methods are used to group test cases based on observed similarities or patterns in the prioritization criteria. This clustering process helps organize test cases into manageable groups, enabling more focused and efficient testing strategies.

To enhance this prioritization process, the framework incorporates artificial intelligence (AI) techniques to assess the accuracy or defect detection rates associated with the prioritized test cases. Machine learning models or neural networks may be used to analyze the prioritized cases and predict the likelihood of defects or failures within the software system.

The integration of fuzzy logic, clustering algorithms, and AI within this framework aims to improve the precision and effectiveness of test case prioritization. By considering multiple parameters, using fuzzy logic for initial prioritization, grouping test cases with clustering techniques, and applying AI for defect prediction, the framework seeks to optimize the testing process, facilitating early fault detection and enhancing overall software quality.

Test case prioritization using machine learning encompasses several critical stages: data collection and preparation, feature selection, data splitting into training and testing sets, algorithm selection and configuration, model training and evaluation, performance optimization, and final ranking and prioritization of test cases. Each stage is vital for effectively identifying and prioritizing the most crucial test cases, thereby improving the efficiency and effectiveness of software testing.

K-Means clustering plays a key role by partitioning test cases into distinct groups based on their similarities. This technique groups test cases with similar features into clusters, ensuring that cases within the same cluster are more alike than those in other clusters. This method aids in prioritizing test cases by focusing on representative cases from each cluster, which enhances diverse and thorough test coverage. Consequently, this approach allows for efficient resource allocation by prioritizing the most significant clusters first.

Random Forest is an ensemble learning technique that builds multiple decision trees and combines their outputs to make a final prediction. In the context of test case prioritization, Random Forest can evaluate the significance of different test cases based on their features. By aggregating the predictions from numerous decision trees, Random Forest determines which test cases are most likely to have a substantial impact or uncover critical defects. This approach improves prioritization by harnessing the collective insights of multiple trees to effectively assess and rank test cases.

Upon finalizing the comparison and prioritization of test cases, the framework employs clustering to group similar cases, such as combining the top five into a single cluster. This step is key in organizing the test cases into coherent and manageable sets, allowing for improved efficiency in both execution and analysis. Clustering helps to detect patterns, minimize redundancy, and streamline the overall testing process.

Various clustering techniques can be used, including hierarchical, density-based, and partitioning methods. K-means clustering is particularly suited to this framework due to its simplicity, scalability, and efficiency. It groups data into a predetermined number of clusters, reducing variance within each group, and ensuring test cases in the same cluster share the highest degree of similarity based on prioritization metrics.

K-means is selected for its capacity to efficiently manage large datasets, its ease of implementation, and its flexibility in accommodating different data distributions. By utilizing K-means clustering, the framework not only groups test cases into relevant categories but also boosts the overall efficiency of the testing process. This method ensures that the most important clusters receive priority, enhancing the effectiveness of the testing strategy.

The APFD (Average Percentage of Faults Detected) metric is a well-established tool for evaluating the effectiveness of test case prioritization strategies by measuring how quickly faults are uncovered during testing. By analyzing the APFD scores, we can gauge the efficiency of our prioritization method. A higher APFD score for prioritized test cases signifies that the approach successfully identifies faults earlier in the testing process, allowing for quicker detection and resolution of issues. In contrast, non-prioritized test cases generally show a lower APFD score, indicating delayed fault detection, which can slow down the overall testing progress.

Results

To demonstrate the framework's effectiveness, the study evaluates its performance across four datasets of varying sizes:

- 8 test cases from the triangle problem (small dataset),
- 87 test cases from the healthcare domain (small dataset),
- 5,000 test cases from the manufacturing domain (medium dataset),

- 19,200 test cases from the banking domain (large dataset).

This evaluation emphasizes the scalability and robustness of the framework, illustrating its capacity to handle datasets of different sizes. For smaller datasets, the research focuses on how effectively the framework prioritizes and clusters test cases. Moreover, the APFD (Average Percentage of Faults Detected) rates for both prioritized and non-prioritized test cases are compared to highlight improvements in fault detection efficiency, showing the framework's ability to enhance early fault detection.

Table 1 Comparative Analysis of Decision Tree, Random Forest & Neural Network

Algorithm	Number of Test Cases	Fault Detection Rate	Execution Time
Decision Tree	5000	97.0%	15seconds
Random Forest	5000	96.2%	30seconds
Neural Network	5000	99.4%	58seconds

Table 2 : Comparative Analysis of Decision Tree, Random Forest & Neural Network

Algorithm	Number of Test Cases	Fault Detection Rate	Execution Time
Decision Tree	19200	97.8%	21seconds
Random Forest	19200	99%	40seconds
Neural Network	19200	99.7%	78seconds

Based on the results shown in Table 1 & 2, the application of optimization techniques led to significant performance improvements for various machine learning algorithms in test case prioritization:

The optimization strategies applied to these algorithms led to improvements in key performance metrics. The goal was to balance reducing the number of test cases with maintaining a high fault detection rate. Each algorithm responded differently to optimization, with varying effects on execution time. These findings underscore the potential of optimizing machine learning algorithms for more efficient and effective test case prioritization in software testing.

The research paper's findings demonstrate that machine learning algorithms can be highly effective for test case prioritization, as shown in Table 1& 2. Various metrics were used to assess performance, including test case execution count, fault detection rate, and execution time. The results revealed that machine learning approaches surpassed traditional methods, such as code coverage and risk-based techniques, particularly in terms of fault detection rate and test case efficiency. Notably, the neural network emerged as the top performer, achieving the highest fault detection rate with the fewest test cases executed.

However, the study also acknowledged some limitations of machine learning for test case prioritization, such as the need for extensive, high-quality datasets and the risk of overfitting. Future work could address these challenges by experimenting with alternative machine learning algorithms and incorporating more advanced data preprocessing methods. Overall, the research illustrates the potential of machine learning to significantly improve the efficiency and effectiveness of software testing.

• The neural network algorithm detected 99.7% of the faults, the highest rate among the tested methods. However, it had the longest execution time and executed the least number of test cases.

• The decision tree algorithm, while completing execution in the shortest amount of time and processing the most test cases, detected only 97.8% of faults, the lowest detection rate.

• The random forest algorithm offered a middle ground, balancing fault detection rate, execution time, and the number of test cases run.

These findings highlight that each machine learning algorithm has its own advantages and drawbacks when it comes to test case prioritization. The choice of algorithm should be aligned with the specific goals and limitations of the task at hand. It is also important to remember that these results are specific to the dataset and problem used in this study and may not apply universally. Evaluating the performance of multiple algorithms for a particular problem is essential to determine the most suitable approach.

Conclusion

In conclusion, employing machine learning algorithms for test case prioritization greatly enhances the efficiency and effectiveness of software testing. By evaluating various algorithms on a real-world dataset, the study found that the performance of each algorithm varied depending on the dataset's characteristics. The mathematical model developed in this research offers a structured approach to test case prioritization, helping to determine the optimal execution order of test cases. Utilizing linear programming techniques ensures that the solutions generated are both feasible and optimal. Overall, this research emphasizes the potential advantages of integrating machine learning and mathematical modeling techniques in software testing, offering valuable insights for both practitioners and researchers into effective strategies for test case prioritization.

Future Work

Here are several promising directions for future research in test case prioritization using machine learning and mathematical modeling techniques:

1. Advanced Machine Learning Approaches: Investigate the application of advanced machine learning techniques, such as deep learning or reinforcement learning, to enhance the accuracy of test case prioritization.

2. Feature Selection and Weighting: Examine the impact of various feature selection methods and attribute weighting schemes on the performance of machine learning-based prioritization.

3. Enhanced Mathematical Models: Develop more intricate mathematical models that account for factors like resource limitations and dynamic changes in the testing environment.

4. Integration with Other Testing Techniques: Assess how test case prioritization can be combined with other software testing methodologies, such as mutation testing or fault localization, to improve overall testing effectiveness.

5. Applicability Across Software Types: Apply and evaluate the proposed prioritization techniques on different software systems, including mobile applications and web applications, to assess their generalizability and effectiveness.

References

1. Abid, R., & Nadeem, A. (2017). A novel approach to multiple criteria-based test case prioritization. Proceedings of the 2017 13th International Conference on Emerging Technologies (ICET), Islamabad, Pakistan, 27–28 December 2017, 1-6. IEEE.

2. Khatibsyarbini, M., Isa, M. A., Jawawi, D. N. A., & Tumeng, R. (2018). Test case prioritization approaches in regression testing: A systematic literature review. Information and Software Technology, 93, 74–93.

3. Ammar, A., Gaber, M., & Ibrahim, N. (2016). Enhanced weighted method for test case prioritization in regression testing using unique priority value. Proceedings of the 2016 International Conference on Information Science and Security (ICISS), Pattaya, Thailand, 19–22 December 2016, 1-5. IEEE.

4. Konsaard, P., & Ramingwong, L. (2015). Using artificial bee colony for code coverage-based test suite prioritization. Proceedings of the 2015 2nd International Conference on Information Science and Security (ICISS), Seoul, Korea, 14–16 December 2015, 1-4. IEEE.

5. Rosero, R. H., Cruz, J. S., & López, M. A. (2017). Regression testing of database applications under an incremental software development setting. IEEE Access, 5, 18419–18428.

6. Zhu, Q., Wang, H., & Zhang, X. (2018). Test case prioritization using k-means clustering. Journal of Systems and Software, 137, 46–57.

7. Wang, S., Wu, L., & Zhou, P. (2017). Test case prioritization using fuzzy logic based on risk, code complexity, and execution time. Journal of Software Testing, Verification & Reliability, 27(6), 1-16.

8. Srikanth, H., Williams, L., & Osborne, J. (2005). A machine learning approach to test case prioritization for regression testing. Proceedings of the International Symposium on Software Reliability Engineering (ISSRE), 1-10. IEEE.

9. Jiang, H., Chan, K. Y., & Wu, Y. (2016). Fuzzy logic and neural network-based hybrid approach for test case prioritization. International Journal of Software Engineering and Knowledge Engineering, 26(4), 589–610.

10. Zheng, Y., Li, Z., & Liu, X. (2019). K-means clustering with fuzzy logic for test case prioritization. International Journal of Software Engineering and Applications, 10(2), 37–50.

11. Kim DH. Artificial intelligence-based modeling mechanisms for material analysis and discovery. Journal of Intelligent Pervasive and Soft Computing 2022; 1(1): 10–15.

12. Qayyum F, Kim DH, Bong SJ, et al. A survey of datasets, preprocessing, modeling mechanisms, and simulation tools based on AI for material analysis and discovery. Materials 2022; 15(4): 1428.

13. Zaman U, Mehmood F, Iqbal N, et al. Towards secure and intelligent internet of health things: A survey of enabling technologies and applications. Electronics 2022; 11(12): 1893. doi: 10.3390/electronics11121893

14. Meriem, A., & Abdelaziz, M. (2019, March). A Methodology to do Model-Based Testing using FMEA. In Proceedings of the 2nd International Conference on Networking, Information Systems & Security (pp. 1-11).

15. Srivastava, H., & Gupta, A. (2019). Test case prioritization using K-means clustering. Proceedings of the 2019 IEEE International Conference on Information Communication and Computing Systems (ICICCS), 73–78.

16. Agarwal, D. S., & Sharma, M. (2020). Enhancing test case prioritization using clustering techniques. Journal of Systems and Software, 159, 110.

17. Bansal, M., & Gupta, S. (2021). A novel approach for test case prioritization using clustering algorithms. Proceedings of the 2021 IEEE International Conference on Robotics and Automation for Humanitarian Applications (ICRAI), 451–458.

18. Singh, A. R., & Kumar, N. (2020). Regression test case prioritization using agglomerative clustering. Procedia Computer Science, 171, 72–81.