

MACHINE LEARNING

An Adaptive Risk-Aware Machine Learning Framework for Cost-Effective Regression Test Case Prioritization

Priya Sawant, Jyoti Tale

Department of Information Technology, Siddhant College of Engineering, Sudumbare, Pune, Maharashtra, India

Published October 2026

Abstract

Regression testing is essential for identifying defects after software changes, but running all test cases can be time-consuming and costly. This paper proposes an Adaptive Risk-Aware Machine Learning Framework for Cost-Effective Regression Test Case Prioritization. The framework uses a Random Forest classifier to predict test case risk based on change impact, historical failures, code coverage, execution cost, and recent results. It dynamically prioritizes test cases using predicted risk, execution cost, and feedback from prior executions. We evaluated the approach using APFD, APFDc, fault detection rate, and execution cost. Experimental results show an APFD of 0.7173 and an APFDc of 0.7987, demonstrating improved test case prioritization compared with the selected baseline techniques.

Keywords: adaptive testing; cost-effective testing; machine learning; regression testing; test case prioritization

1 Introduction

Software testing is an essential phase of the software development life cycle that helps ensure the quality, reliability, and correctness of software applications. During software maintenance and enhancement, developers frequently modify existing code to introduce new features, correct defects, or improve system performance. Although these changes are necessary, they may unintentionally affect previously functioning components and introduce new defects. Regression testing is performed to verify that software modifications do not negatively affect existing functionality. However, repeatedly executing the complete regression test suite can require considerable time, computational resources, and testing effort, particularly in large and frequently updated software systems.

Regression Test Case Prioritization (TCP) is an effective technique for addressing this challenge by arranging test cases in an order that allows important faults to be detected as early as possible. Traditional prioritization approaches commonly consider factors such as code coverage, historical test failures, or test execution cost. Although these techniques can improve testing efficiency, individual criteria may not adequately represent the combined risk and cost associated with each test case. For example, a test case covering recently modified code may be important even if it has a low historical failure count. Similarly, a high-risk test case may be expensive to execute, making the balance between early fault detection and testing cost an important consideration.

Machine learning provides opportunities to improve test case prioritization by identifying patterns in historical testing data and estimating the likelihood of fault detection. By considering multiple features together, a machine learning model can support more informed prioritization decisions than

a strategy based on a single criterion. However, an effective prioritization framework should also consider execution cost and respond to newly observed test results so that priorities can be adjusted as testing information changes.

This paper presents an Adaptive Risk-Aware Machine Learning Framework for Cost-Effective Regression Test Case Prioritization. The proposed framework considers software change impact, historical failures, code coverage, execution cost, and recent test execution results as input features for risk prediction. A Random Forest classifier is used to estimate the fault-detection risk of individual test cases. The predicted risk is then combined with cost-aware ranking and execution feedback to support adaptive prioritization. The framework is implemented using a Python-based FastAPI backend and a React frontend, providing an interface for viewing prioritized test cases and their associated risk information.

The proposed approach is evaluated using a dataset containing 100 regression test cases from six software modules. Its performance is compared with coverage-based, historical-failure-based, and execution-cost-based prioritization approaches. Evaluation metrics include Average Percentage of Faults Detected (APFD), cost-cognizant Average Percentage of Faults Detected (APFDc), fault-detection rate, and execution cost. The objective is to investigate whether combining machine learning-based risk prediction, execution-cost consideration, and adaptive feedback can improve fault-detection effectiveness while reducing the cost of regression testing.

The remainder of this paper is organized as follows. Machine learning-based regression test case prioritization is discussed first, followed by the proposed methodology. The experimental results are then presented and discussed, and the final section concludes the paper and identifies potential directions for future work.

2 Machine learning-based regression test case prioritization

Machine Learning (ML) is a branch of artificial intelligence that enables computer systems to learn patterns from data and use them to make predictions or decisions. In software testing, machine learning techniques can analyze historical test execution results, software characteristics, and defect information to support more effective testing strategies. Regression testing is particularly suitable for such techniques because test suites often contain a large number of test cases that must be executed repeatedly after software modifications. Machine learning-based Test Case Prioritization (TCP) aims to arrange test cases according to their predicted importance or likelihood of detecting faults, enabling critical defects to be identified earlier in the testing process.

Traditional regression test case prioritization techniques generally rely on criteria such as code coverage, historical failure information, execution cost, or modification impact. Although these techniques can improve testing efficiency, a method based on a single criterion may not capture the combined influence of multiple factors. For example, a test case associated with a recently modified software component and a history of failures may require higher priority. However, its execution cost must also be considered when the objective is to identify faults efficiently within limited testing resources. Machine learning can address this challenge by analyzing multiple features together and estimating the relative risk associated with individual test cases.

Several machine learning algorithms, including Decision Trees, Random Forest, Support Vector Machines, and Logistic Regression, can be applied to software testing and defect prediction problems. Among these techniques, Random Forest is an ensemble learning algorithm that combines the predic-

tions of multiple decision trees to produce a final prediction. It can model relationships between multiple input features and is suitable for classification tasks involving structured datasets. In the proposed framework, a Random Forest classifier is used to estimate the fault-detection risk of regression test cases using software change impact, historical failures, code coverage, execution cost, and recent execution results.

The application of machine learning to regression test case prioritization also supports the integration of risk-based and cost-aware testing strategies. Risk prediction helps identify test cases that may be important for detecting faults, while execution-cost information supports decisions about the order in which tests should be executed. By combining these factors, the framework aims to balance early fault detection with efficient utilization of testing resources. Performance can be assessed using measures such as Average Percentage of Faults Detected (APFD), cost-cognizant Average Percentage of Faults Detected (APFDc), fault-detection rate, and total execution cost.

An additional challenge in regression testing is that test case importance may change as new software modifications and execution results become available. A fixed prioritization order may not reflect updated testing conditions. Adaptive prioritization addresses this limitation by incorporating execution feedback and updating relevant information used to determine subsequent priorities. In the proposed framework, newly observed test results and historical failure information support the adaptation of test case rankings as testing progresses.

The proposed research combines machine learning-based risk prediction, cost-aware ranking, and adaptive feedback within a unified regression test case prioritization framework. The framework is implemented using a Python-based FastAPI backend and a React frontend. It is evaluated on a dataset of 100 regression test cases distributed across six software modules and compared with coverage-based, historical-failure-based, and execution-cost-based approaches. The objective is to improve fault-detection effectiveness while reducing testing effort and execution cost. This approach provides a practical way to investigate how predictive analysis and adaptive prioritization can contribute to more efficient regression testing.

3 Proposed methodology

The proposed methodology presents an adaptive risk-aware machine learning framework for cost-effective regression test case prioritization. The framework aims to identify test cases with a higher likelihood of detecting faults and arrange them in an appropriate execution order while considering testing cost. It combines historical test information, software change impact, code coverage, machine learning-based risk prediction, and execution feedback to improve prioritization decisions. The methodology consists of the following stages.

3.1 Dataset collection

The first stage involves collecting regression test case data containing relevant information about software modules and test execution. The dataset used in this study contains 100 test cases distributed across six modules: Authentication, Cart, Orders, Payments, Products, and Users. Each test case includes attributes such as change impact, historical failures, code coverage, execution cost, recent execution result, and fault-detection status. These attributes provide the information required to train the machine learning model and determine test case priorities.

3.2 Data preprocessing and feature selection

In this stage, the collected dataset is examined and prepared for machine learning. The input attributes are selected according to their relevance to fault-detection risk and execution efficiency. Recent execution results are converted into numerical values, with PASSED represented as 0 and FAILED represented as 1. The selected input features are change impact, historical failures, code coverage, execution cost, and recent execution result. The fault-detection status is used as the target variable for classification.

3.3 Machine learning-based risk prediction

A Random Forest classifier is employed to estimate the fault-detection risk of each regression test case. The model learns patterns from the selected features and predicts the probability associated with fault detection. The dataset is divided into training and testing subsets using an 80:20 split, with stratification to maintain the distribution of target classes. The trained model is subsequently used to generate risk predictions for test cases. These predictions provide an important input for determining the priority of each test case.

3.4 Cost-aware test case prioritization

After risk prediction, the test cases are ranked by combining the predicted risk with execution-cost considerations. Test cases with higher predicted risk are considered for earlier execution, while execution cost is incorporated to support efficient use of testing resources. This stage aims to balance the potential for early fault detection with the effort required to execute the test suite. The resulting prioritized list provides the order in which test cases should be considered for regression testing.

3.5 Adaptive feedback mechanism

The adaptive mechanism incorporates newly observed execution results into the prioritization process. When a test case is executed, its result can be used to update relevant information, including recent execution status and historical failure information. These updated values can influence subsequent risk predictions and rankings. Consequently, the framework can respond to changes in testing information rather than relying exclusively on a fixed prioritization order.

3.6 Performance evaluation

The final stage evaluates the effectiveness of the proposed framework by comparing it with coverage-based, historical-failure-based, and execution-cost-based prioritization approaches. Average Percentage of Faults Detected (APFD), cost-cognizant Average Percentage of Faults Detected (APFDc), fault-detection rate, and execution cost are used as evaluation measures. The results are analyzed to determine whether the combination of machine learning, cost-aware ranking, and adaptive feedback improves prioritization effectiveness and supports cost-effective regression testing.

4 Experimental results and discussion

The proposed adaptive risk-aware machine learning framework was evaluated to investigate its effectiveness in regression test case prioritization. The experimental analysis compares the proposed approach with coverage-based, historical-failure-based, and execution-cost-based prioritization techniques. The evaluation focuses on fault-detection effectiveness and the cost associated with test

execution. APFD, APFDc, fault-detection rate, and execution cost are used to assess the performance of the different approaches.

4.1 Experimental setup

The dataset consists of 100 regression test cases distributed across six software modules. Five input features—change impact, historical failures, code coverage, execution cost, and recent execution result—are used by a Random Forest classifier to predict fault-detection risk. The dataset is divided into training and testing subsets using an 80:20 stratified split. The trained model generates risk predictions, which are used by the proposed framework to prioritize test cases while considering execution cost and adaptive feedback.

4.2 Performance comparison

The performance of the proposed framework is compared with three baseline techniques. The results obtained from the experimental evaluation are presented in Table 1.

Table 1. Performance comparison of prioritization techniques.

Prioritization technique	APFD	APFDc	Fault detection rate (%)	Execution cost
Coverage-based	0.5118	0.5088	44.0	1062.44
Historical-failure-based	0.5184	0.5104	44.0	1062.44
Execution-cost-based	0.4989	0.6501	44.0	1062.44
Proposed adaptive risk-aware ML	0.7173	0.7987	44.0	1047.89

4.3 Analysis of results

The experimental results indicate that the proposed adaptive risk-aware machine learning approach achieves an APFD value of 0.7173, which is higher than the values obtained using coverage-based, historical-failure-based, and execution-cost-based prioritization. This result indicates improved fault-detection effectiveness according to the APFD measure.

The proposed framework also achieves an APFDc value of 0.7987, outperforming the three baseline approaches in the reported evaluation. This indicates better cost-cognizant fault-detection performance under the experimental conditions. The execution cost of the proposed approach is 1047.89, compared with 1062.44 for each baseline technique, representing a reduction of approximately 1.37%.

The reported fault-detection rate is 44.0% for all four approaches. Therefore, the observed differences are in the ordering and cost-related performance measures rather than in the reported overall fault-detection rate. The findings suggest that combining machine learning-based risk prediction, cost-aware prioritization, and adaptive feedback can improve the order in which test cases are executed while modestly reducing the recorded execution cost.

4.4 Discussion

The results demonstrate the potential benefits of using multiple test case characteristics to determine execution priorities. Unlike approaches that rely on a single criterion, the proposed framework considers software change impact, historical failures, code coverage, execution cost, and recent

execution results when estimating risk. Incorporating feedback further supports updating priorities as testing information changes.

However, these findings are based on a dataset of 100 test cases, and further evaluation using larger datasets and additional software projects would be necessary to establish the generalizability of the approach. The reported results provide an initial experimental basis for assessing the effectiveness of adaptive risk-aware regression test case prioritization.

5 Conclusion

This research presents an Adaptive Risk-Aware Machine Learning Framework for Cost-Effective Regression Test Case Prioritization. The proposed framework addresses the challenges of executing large regression test suites by combining machine learning-based risk prediction, execution-cost consideration, and adaptive feedback. A Random Forest classifier is used to estimate fault-detection risk based on software change impact, historical failures, code coverage, execution cost, and recent test execution results. The predicted risk is then used to support the prioritization of test cases, while execution feedback helps update relevant information for subsequent prioritization decisions.

The framework was implemented using a Python-based FastAPI backend and a React frontend and evaluated on a dataset of 100 regression test cases distributed across six software modules. Its performance was compared with coverage-based, historical-failure-based, and execution-cost-based prioritization techniques. The proposed approach achieved an APFD value of 0.7173 and an APFDc value of 0.7987, with a recorded execution cost of 1047.89. Compared with the baseline execution cost of 1062.44, this represents a reduction of approximately 1.37%. These results indicate improved performance in the reported APFD and APFDc measures, together with a modest reduction in execution cost.

The study demonstrates the potential of combining multiple test case characteristics to support more effective regression test case prioritization. However, the evaluation is limited to a relatively small dataset, and further testing on larger datasets and real-world software projects is required to assess the generalizability of the framework. Future work may explore additional machine learning algorithms, more comprehensive cost models, and advanced feedback mechanisms to improve risk estimation and prioritization. Overall, the proposed framework provides a practical foundation for investigating adaptive, risk-aware, and cost-effective regression testing.

Acknowledgment

The authors would like to express their sincere gratitude to their project guide, Prof. Jyoti Tale, for her valuable guidance, continuous support, and constructive suggestions throughout this research work. The authors are also thankful to the Department of Information Technology, Siddhant College of Engineering, Sudumbare, Maharashtra, for providing the necessary academic support and encouragement to carry out this research.

Finally, the authors extend their appreciation to everyone who contributed directly or indirectly to the successful completion of this research paper.

References

- [1] Elbaum S, Malishevsky AG, Rothermel G. Test Case Prioritization: A Family of Empirical Studies. *IEEE Transactions on Software Engineering*. 2002;28(2):159–182. <https://doi.org/10.1109/32.988497>

-
- [2] Rothermel G, Untch RH, Chu C, Harrold MJ. Prioritizing Test Cases for Regression Testing. *IEEE Transactions on Software Engineering*. 2001;27(10):929–948. <https://doi.org/10.1109/32.962562>
 - [3] Huang YC, Huang CY, Chang JR, Chen TY. Design and Analysis of Cost-Cognizant Test Case Prioritization Using Genetic Algorithm with Test History. In: *Proceedings of the 2010 IEEE 34th Annual Computer Software and Applications Conference (COMPSAC), Seoul, South Korea*. 2010. p. 413–418. <https://doi.org/10.1109/COMPSAC.2010.66>
 - [4] Pan R, Bagherzadeh M, Ghaleb TA, Briand LC. Test Case Selection and Prioritization Using Machine Learning: A Systematic Literature Review. *Empirical Software Engineering*. 2022;27(2):29. <https://doi.org/10.1007/s10664-021-10066-6>
 - [5] Breiman L. Random Forests. *Machine Learning*. 2001;45:5–32. <https://doi.org/10.1023/A:1010933404324>
 - [6] Elbaum S, Malishevsky AG, Rothermel G. Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization. In: *Proceedings of the 23rd International Conference on Software Engineering (ICSE)*. 2001.