DOI : https://doi.org/10.56452/7-8-45

ISSN: 0974-5823 Vol. 7 No. 8 August, 2022

International Journal of Mechanical Engineering

Prioritization of Test Cases using Machine Learning: A Novel Approach

[1]Tapas Kumar Choudhury, [2] Subhendu Kumar Pani, [3]Jibitesh Mishra

[tkchoudhury24, skpani.india, mishrajibitesh]@gmail.com

[1]Computer Science & Engineering, Biju Patnaik University of Technology, Odisha, India

[2]Krupajal Computer Academy, Bhubaneswar, Odisha, India

[3]Odisha University of Technology and Research, Bhubaneswar, Odisha, India

Abstract

Test case prioritization is a critical process in software testing that involves determining the order in which test cases should be executed. The goal is to identify high-priority test cases that have a higher likelihood of finding defects early in the testing process. Machine learning has been a valuable tool that helps automate the process of prioritizing test cases based on various factors. In this paper, the priority of a business requirement, complexity of the associated testcases, and estimated time and cost of a testcase are considered for prioritizing testcases. Using popular machine learning models,it has been found that the approach provides encouraging results.

Keywords: Test Case Prioritization, Machine Learning, Requirements Priority, Testing Process

1. Introduction

The software testing process is crucial for ensuring the quality and reliability of software products.However, it also comes with several challenges that testers and quality assurance professionals need to address.Software requirements often change throughout the development process, which can result in the need to continuously adapt test cases and strategies to align with the evolving specifications.As software systems become increasingly complex, testing all possible scenarios and interactions becomes challenging, leading to potential defects going undetected [9].Testing is often conducted under tight schedules, particularly in agile and fast-paced development environments. Limited time for testing can lead to incomplete coverage and a rush to meet deadlines.Inadequate testing resources, including skilled testers, testing tools, and testing environments, can hinder the thoroughness and effectiveness of the testing process. Thus, incomplete test coverage makes it difficult to ensure that all aspects of the software have been thoroughly tested.

Test case prioritization (TCP) is a crucial aspect of software testing, aiming to optimize testing resources by executing high-impact test cases early in the testing process. The traditional prioritization methods rely on coverage-based or risk-based strategies, which may not fully capture the complexities of modern software systems. In recent years, the integration of machine learning techniques has shown significant promise in enhancing the efficiency and effectiveness of test case prioritization. Machine learning offers the potential to consider diverse factors, such as code changes, historical defect data, code complexity, and

execution time, to predict the likelihood of defects being detected by specific test cases. This predictive capability enables the identification of high-priority test cases that are more likely to uncover critical defects.Several studies have explored the application of various machine learning algorithms, including decision trees, random forests, neural networks, and ensemble methods, for test case prioritization. These algorithms leverage historical testing data to learn patterns and relationships between test cases, code changes, and defect occurrences. As a result, they offer more adaptive and data-driven prioritization strategies that can adapt to the evolving nature of software systems [10].

In this paper, priority of a business requirement and complexity of the associated testcases are given due attention. Considering the time and cost involved in the testcase execution, a prioritization approach has been studied.

2. Related Works

Test case prioritization using machine learning is an active area of research that aims to leverage machine learning techniques to enhance the effectiveness of test case prioritization. Here are select papers and research works that represent a subset of the extensive research conducted in the field of test case prioritization. They highlight different approaches, challenges, and insights into improving testing efficiency and defect detection rates through effective test case prioritization techniques.

Ren et al. [1] proposed a two-layer model for test case prioritization in regression testing of GUI software. While the inner layer is a function call graph, the outer layer is an even handler tree. The approach uses more amount of source code information compared to traditional methods for prioritization and centrality measure is considered for highlighting modified functions in case of change in software version.

Getachew et al. [2] developed a test case prioritization (TCP) technique that uses static software metrics such as McCabe's Cyclomatic complexity, and Halstead's metrics to develop TCP algorithm. The TCP approach focuses on the structural coverage such as method coverage statement coverage and branch coverage. The proposed technique was compared with the existing approaches and found to be efficient.

Chen et al. [3] proposed TCP based on historical execution information to detect higher faults in continuous integration environment. The authors claim that historical execution failure information helps achieve better test case prioritization and outperforms other techniques. Further they find that if historical execution information is insufficient then it can be combined with requirements priority to provide better performance.

Ali Samad et al. [4] proposed multi objective particle swarm optimization (MOPSO) for TCP. The approach considers minimum execution time, maximum code coverage and maximum fault detection ability of the test case. The technique used three known datasets and open-source Java applications for experimental study. Using different evaluation metrics such as inclusiveness, precision and size reduction, the experimental data showed encouraging results when benchmarked with other approaches.

Roza et al. [5] introduced sliding window method for TCP in continuous integration environment. The research work considered Random Forest (RF) and Long Short-Term

Memory (LSTM) deep learning network for prioritization and code analysis. Experiment was conducted on eleven systems and found that the Random Forest algorithm produced best performance compared to that found in literature.

Dahiya and Solanki [6] proposed a novel requirement-based test case prioritization method for regression testing. It uses the natured inspired Cuckoo search optimization with support vector machine (SVM). The method was evaluated with Firefly algorithm and found to be improved one in terms of average percentage of fault detection and reduced execution time.

Waqar et al. [7] proposed a four-step method based on reinforcement learning for TCP. In the first step, a log dataset of users' and testers' interaction with the system was prepared, in the second step, the proposed reinforcement algorithm was used to predict the reward sequence. Then the proposed prioritization algorithm prioritized the test suit, and finally it was validated by test experts. Considering five case study applications, the performance evaluation results show that the proposed method outperforms the baseline approaches. The method was evaluated using five case study applications and found to outperform the baseline approaches.

Rezwana Mamata [8] studied the potential of transfer learning to improve the performance of test case prioritization and failure of test case in software projects involving continuous integration environments. It proposed a technique called TCP-TB to prioritize test cases using prediction probability of test failures generated by a transfer learning algorithm. The study found that transfer learning has been the most effective compared to other machine learning approaches.

3. Rationale of the Approach

Testcase prioritization based on business requirements, complexity of associated test cases, in addition to the cost, and time of test execution is a holistic approach that considers both the criticality of business functionalities and the practical aspects of resource allocation. It ensures that testcases addressing critical business functions and complex scenarios are executed earlier in the testing process. This approach enhances the overall quality of the software by focusing on areas that have higher potential impact or are more likely to contain defects.This approach aims to strike a balance between testing effectiveness, resource utilization, and business requirements. The process is depicted in Figure-1 and described further here.

a) Requirement Analysis and Categorization: Understand the business requirements thoroughly and categorize them based on their criticality to the application's functionality. Assign priority levels to different business requirements, considering factors such as user impact, regulatory compliance, revenue generation, and customer satisfaction.

b) Complexity Assessment: Evaluate the complexity of individual test cases by considering factors like data combinations, business rules, integration points, and user interactions. More complex scenarios are likely to have a higher likelihood of defects and should be given priority.

c) Time and Cost Estimation: Estimate the cost of executing each test case considering factors such as manual effort, automation effort, required resources, and infrastructure.Estimate the time required for executing each test case, considering factors like test duration, setup time, and teardown time.

DOI : https://doi.org/10.56452/7-8-45

d) Test Case Prioritization: Categorize a test case based on the business requirements it covers and the complexity level of the test case, associated cost, and time of test execution. Assign priority to a test case based on the combination of these factors.

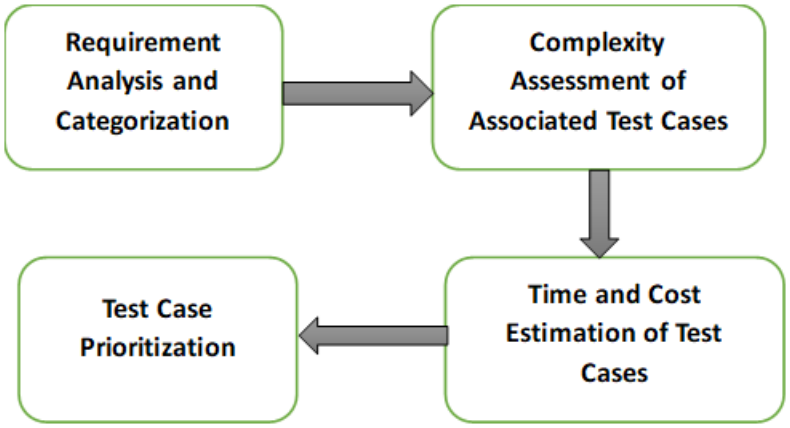

Figure-1: Test Case Prioritization Process

Since, manual process of test case prioritization is difficult and time consuming, it is recommended to exploits supervised machine learning techniques to automate the process of prioritization of the test cases and accrue the effectiveness thereof.

4. Experiment Design

The novel approach described in section-3 has been implemented leveraging the supervised machine learning techniques. Weka, a popular machine learning framework has been used to conduct experiments [11].

4.1 Dataset

The dataset considered in the research has been taken from the Kaggle open-source repository [12]. The dataset has been collected from a real-world software project as informed by the contributor. It contains 2000 instances and six predictive attributes. The attributes are described in Table-1 and a snapshot of the data is provided in Figure-1.

Table-1: Dataset Descriptions

Attribute Name	Data Type	Description
B_Req	Numeric	Business Requirements
R_Priority	Numeric	Requirement Priority of business requirement
FP	Categorical	Function point of each testing task; in this case test cases against each requirement that covers a particular FP
Complexity	Numeric	Complexity of a particular function point
Time	Numeric	Estimated max time assigned to each Function Point of testing task by QA team lead
Cost	Numeric	Calculated cost for each function point using complexity and time with function point estimation technique to calculate cost
Priority	Categorical	The Class Attribute with values Low, Medium, High; is the assigned testcases priority against each Function Point

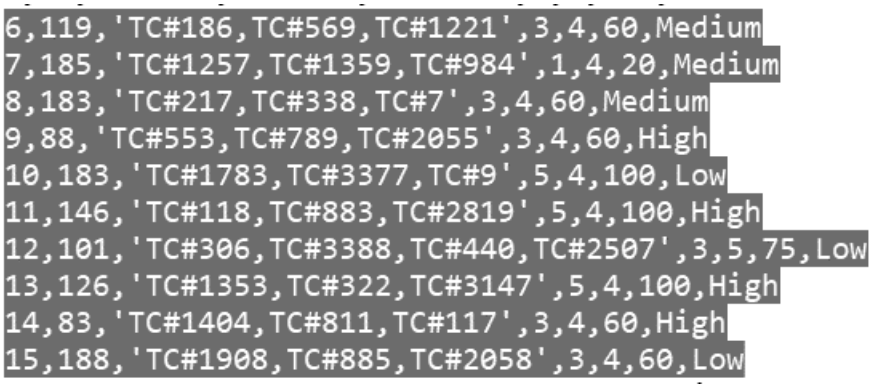

Figure-2: Snapshot of the Dataset

4.2 Data Preprocessing and Candidate ML Algorithms

The attribute selection filter found in Weka has been used along with Gain-Ratio attribute evaluator algorithm and a ranker algorithm to evaluate and rank the attributes based on their predictive power to predict the priority of a test case. The attribute B_Req that represents business requirements has been ranked lowest. Since it is a serial number that identifies business requirements has been removed.

The dataset contained numeric variables with data in different ranges. Feature scaling using min-max normalization has been used to transform the data to fall in a predefined range of 0 and 1. The min-max normalization has been used using the following formula:

$$Y' = \frac{Y - Y_{min}}{Y_{max} - Y_{min}} \times (max_{new} - min_{new}) + min_{new}$$

The initial value to be normalized is Y.$Y_{min}$and Y_{max} represent the lowest value, and the highest value found in the dataset. The normalized value is Y ′, and the intended minimum and maximum values of the normalized range are min_{new} and max_{new}, respectively.

Popular machine learning algorithms such as support vector machine, logistic regression, naïve bayes, k-nearest neighbor, decision tree and random forest which have been used by other researchers and found in the literature have been considered as candidate ML algorithms in the study.

Using 5-fold cross validation as the test option, the algorithms have been run on the preprocessed dataset to generate the experimental outputs. The algorithms have been configured with the default parameters and hyperparameters to generate baseline models.

5. Results and Discussion

The experimental data has been tabulated in Table-2 and show in Figure-3 to compare the performance of the machine learning algorithms. Since accuracy and error measure the classification model's efficacy with a premise that a model should achieve higher accuracy with lower computational error, accuracy percentage and root mean squared error (RMSE) have been considered to for comparative study of the machine learning models.

DOI : https://doi.org/10.56452/7-8-45

Table-2: Comparative performance of ML Models

ML Model	Accuracy %	RMSE
Support Vector Machine	84.50	0.541
Logistic Regression	84.45	0.472
Naïve Bayes	85.70	0.472
k-Nearest Neighbor	83.05	0.622
J48 Decision Tree	84.05	0.471
Random Forest	85.25	0.472

The experimental results clearly show that all the models compete in terms of the evaluation parameters. While k_Nearest Neighbor perform the lowest among the models at hand, Naïve Bayes shows the best performance. The RMSE of both Naïve Bayes and Random Forest are same but the accuracy of Naïve Bayes is higher than that of Random Forest. Thus, Naïve Bayes model is recommended for classification in the context.

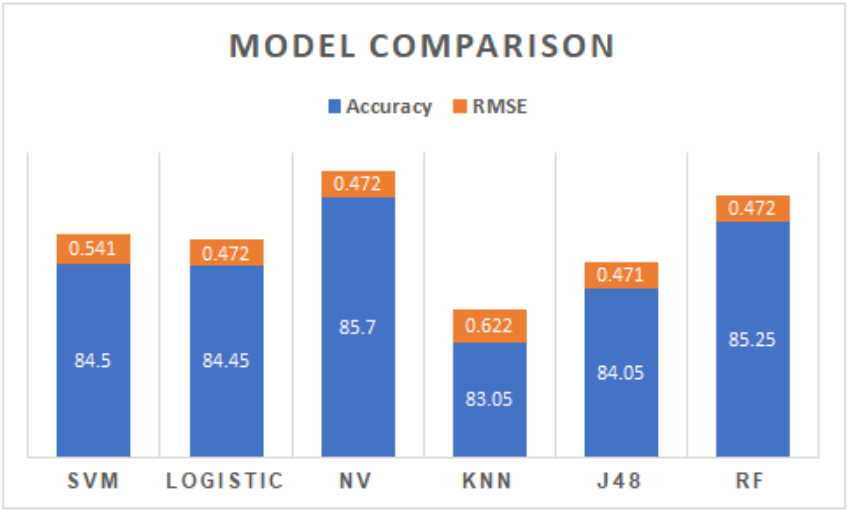

Figure-3: Model Performance Comparison

6. Conclusions

Since exhaustive test is far from practice, test case prioritization is an economic viability to achieve reliable software. Test case prioritization based on relevant criteria achieves a high rate of defect detection and reduces the cost and time of test execution. Machine learning algorithms play a major role towards this end. In this research, business requirements priority has been considered as the primary basis of the test case prioritization due to its importance in software development. Further, business requirements or software requirements keep on changing in an agile development environment making the testing process challenging. The research study finds that machine learning models can effectively prioritize test cases and help achieve reliable and economical software.

References

[1] Y. Ren, B. -B. Yin and B. Wang, (2018). "Test Case Prioritization for GUI Regression Testing Based on Centrality Measures," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 2018, pp. 454-459, doi: 10.1109/COMPSAC.2018.10275.

[2] Getachew, D., Mohapatra, S.K., Mohanty, S. (2022). A Heuristic-Based Test Case Prioritization Algorithm Using Static Metrics. In: Khari, M., Mishra, D.B., Acharya, B., Gonzalez Crespo, R. (eds) Optimization of Automated Software Testing Using Meta-Heuristic Techniques. EAI/Springer Innovations in Communication and Computing. Springer, Cham. https://doi.org/10.1007/978-3-031-07297-0_4

[3] R. Chen, Z. Xiao, L. Xiao and Z. Li, (2022). "Regression Testing Prioritization Technique Based on Historical Execution Information," 2022 International Conference on Machine Learning, Cloud Computing and Intelligent Mining (MLCCIM), Xiamen, China, 2022, pp. 276-281, doi: 10.1109/MLCCIM55934.2022.00054.

[4] Ali Samad, Hairulnizam Bin Mahdin, Rafaqat Kazmi, Rosziati Ibrahim, Zirawani Baharum, (2021). "Multiobjective Test Case Prioritization Using Test Case Effectiveness: Multicriteria Scoring Method", Scientific Programming, vol. 2021, Article ID 9988987, 13 pages, 2021. https://doi.org/10.1155/2021/9988987

[5] E. A. Da Roza, J. A. P. Lima, R. C. Silva and S. R. Vergilio, (2022). "Machine Learning Regression Techniques for Test Case Prioritization in Continuous Integration Environment," 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Honolulu, HI, USA, 2022, pp. 196-206, doi: 10.1109/SANER53432.2022.00034.

[6] Omdev Dahiya, Kamna Solanki. (2021). An Efficient Requirement-based Test Case Prioritization Technique using Optimized TFC-SVM Approach, International Journal of Engineering Trends and Technology, 69(1),5-16.

[7] Waqar, M.; Imran; Zaman, M.A.; Muzammal, M.; Kim, J. (2023). Test Suite Prioritization Based on Optimization Approach Using Reinforcement Learning. Appl. Sci. 2022, 12, 6772. https://doi.org/10.3390/app12136772

[8] Rezwana Mamata, (2023). Test Case Prioritization using Transfer Learning in Continuous Integration Environments, Master Thesis, Department of Electrical, Computer and Software Engineering, Faculty of Engineering and Applied Science, University of Ontario Institute of Technology (Ontario Tech University), Oshawa, Ontario, Canada, April 2023.

[9] Uddin Azeem and Anand Abhineet, (2019). Importance of Software Testing in the Process of Software Development, International Journal for Scientific Research & Development| Vol. 6, Issue 12, 2019 pp.2321-0613.

[10] Muhammad Khatibsyarbini, Mohd Adham Isa, Dayang N.A. Jawawi, Rooster Tumeng,(2018).Test case prioritization approaches in regression testing: A systematic literature review,Information and Software Technology,Volume 93,2018,Pages 74-93,ISSN 0950-5849,https://doi.org/10.1016/j.infsof.2017.08.014.

[11] Weka 3: Machine Learning Software in Java, available at https://www.cs.waikato.ac.nz/ml/weka/.

[12] A test case data set with requirements, https://www.kaggle.com/datasets/zumarkhalid/a-test-case-data-set-with-requirements?resource=download , accessed on 26th July 2023.