International Journal of Sustainable Development Through AI, ML and IoT
Volume 4 | Issue1 | 2025
https://ijsdai.com/index.php/IJSDAI/index
ISSN (Online): 2584-0827

Test Case Prioritization Through Clustering: A Data-Driven Approach

Sheetal Sharma[1], Swati V. Chande[2] & Dr N.K. Joshi[3]

1 Rajasthan Technical University, Kota 324010, India
2 International School of Informatics & Management, Jaipur 302020, India
3 Modi Institute of Management & Technology, KOTA-324009 Rajasthan India
Corresponding author: sheetaljoshi2e@gmail.com

ARTICLE INFO

Received: 1 June 2025
Revised: 17 June 2025
Accepted: 29 July 2025

ABSTRACT

In software development, maintaining software quality and reliability through thorough testing is vital. However, as software systems become increasingly complex, managing large volumes of test cases presents significant challenges [1]. To tackle this problem, effective test case prioritization is necessary to ensure that the most critical tests are executed first [2]. This paper introduces an innovative approach to test case prioritization by combining clustering techniques, specifically K-means, with machine learning algorithms. We explore how K-means clustering can group similar test cases to improve prioritization efficiency [3][4]. Furthermore, we examine the performance of several machine learning models, including Decision Trees (DT), Random Forests (RF), and Neural Networks (NN), comparing their results against traditional methods. The study evaluates these approaches using diverse datasets and metrics such as the number of executed test cases, fault detection rate, and execution time [5]. Experimental findings demonstrate that integrating K-means clustering with machine learning techniques can enhance prioritization by reducing test execution efforts while preserving or even boosting fault detection capabilities. We also highlight the limitations of the proposed method and suggest future research opportunities aimed at further optimizing test case prioritization through advanced machine learning strategies. Overall, this framework offers important contributions toward developing more effective and reliable software testing processes.

.

1. Introduction

The growing complexity of software systems demands comprehensive testing to guarantee their reliability and performance. Regression testing, which ensures that recent changes such as bug fixes and new features do not disrupt existing functionalities, is a vital aspect of this process [1]. To improve the efficiency of regression testing, various techniques are employed, including K-Means selection, minimization, and prioritization. One such technique, K-Means Prioritization (TCP), ranks test cases based on factors like fault detection, fault severity, execution time, and code coverage [6]. This approach helps identify critical faults earlier, enhancing testing efficiency and minimizing potential risks.

The effectiveness of TCP is commonly evaluated using performance metrics such as the Average Percentage of Faults Detected (APFD) and execution time. A higher APFD score indicates quicker fault detection, which contributes to a more efficient testing process [7]. Researchers have also explored clustering techniques to optimize TCP by grouping K-Means with similar characteristics. This clustering enables more systematic prioritization, ensuring that test cases with a higher likelihood of detecting faults are executed first.

Clustering techniques are essential for improving the APFD rate by organizing K-Means based on critical factors: number of faults detected, fault severity and execution time. Studies have shown that clustering algorithms can effectively create distinct groups of K-Means with shared attributes. These organized clusters form the foundation for prioritization, ensuring that K-Means with a higher probability of fault detection are tested earlier [8]. Despite the success of clustering-based approaches in enhancing K-Means prioritization, challenges persist, particularly in developing more robust frameworks and improving the handling of K-Means attributes. Addressing these challenges presents an opportunity for further advancements and refinements in clustering-driven TCP models.

2. Literature Review

A thorough review of existing research highlights significant contributions and developments in the field of test case prioritization (TCP). The foundational work of Leung and White (1989) [1] laid the groundwork for regression testing, stressing the importance of prioritizing K-Means to enhance efficiency and fault detection. Building on this, Rothermel et al. (1999) [9] further explored various TCP strategies, showing how the execution order influences fault detection rates and helps reduce costs.

Following these early contributions, subsequent research delved into different approaches to TCP. Leon and Podgurski (2003) [10] compared coverage-based and distribution-based strategies, analyzing their strengths and weaknesses, particularly when applied to large test suites. Yoo and Harman (2012) [11] highlighted scalability as a significant challenge and proposed optimization techniques such as clustering and fuzzy logic to address this issue. Ahmed and Akbar (2018) [12] introduced a fuzzy logic-based TCP framework to manage uncertainties in prioritization, offering improved efficiency, though it lacked mechanisms to handle redundancy. Chauhan and Bhatnagar (2019) [13] combined fuzzy logic with genetic algorithms to optimize TCP, but their approach did not include clustering techniques. Similarly, Kaur and Singhal (2020) [14] employed fuzzy logic to assign importance to prioritization factors, though they did not investigate the potential benefits of clustering for improved fault detection.

Other studies, such as those by Sharma and Saini (2017) [15] and Yadav and Raghuwanshi (2019) [16], demonstrated the value of fuzzy logic in TCP but noted the need for clustering techniques to better manage large test suites. Meanwhile, machine learning (ML) and deep learning (DL) have introduced innovative approaches to K-Means prioritization. Lachmann et al. (2016) [17] applied Random Forests to predict the importance of K-Means, while Davis and Lee (2019) [18] used DL models to enhance prioritization accuracy, despite facing computational challenges. Pan et al. (2022)

[19] identified scalability and explainability as primary concerns in ML-based TCP approaches, and Swain et al. (2022) [20] integrated metaheuristics with ML to improve prioritization outcomes.
The field of K-Means Prioritization (TCP) has made substantial progress, but there are still numerous areas for improvement. This study provides a comprehensive review of the literature to analyze significant contributions and emerging trends in TCP. Despite advancements, existing TCP techniques continue to face challenges, particularly in managing redundancy and improving execution efficiency [21]. To address these issues, this research proposes a novel framework that integrates clustering techniques with fuzzy logic. By grouping K-Means with similar attributes, clustering can improve execution efficiency, while fuzzy logic helps manage prioritization uncertainties. The proposed framework aims to optimize fault detection and enhance TCP strategies for both small and large-scale test scenarios. A comparative evaluation demonstrates the effectiveness of the framework in improving prioritization and overall software testing efficiency.

3. Proposed Framework

The proposed framework enhances regression testing by combining Fuzzy Logic, Machine Learning (ML), Deep Learning (DL), and K-Means Clustering to optimize test case prioritization. Its main goal is to streamline fault detection while ensuring efficient test execution. Designed to be versatile, the framework is applicable to both small-scale and large-scale test suites, making it relevant across various domains [22][23].
K-Means prioritization is based on three key factors: Number of Faults Detected (NFD), Fault Severity (FS), and Execution Time (ET). The NFD ensures that K-Means likely to detect defects are given higher priority for execution. FS focuses on identifying and addressing critical faults that could significantly affect the software's functionality. ET considers the time needed to execute the tests, balancing the effectiveness of fault detection with efficiency [24][25][26]. These factors serve as inputs to the Fuzzy Logic Controller (FLC), which provides a structured approach to prioritization.
The proposed framework is depicted below:

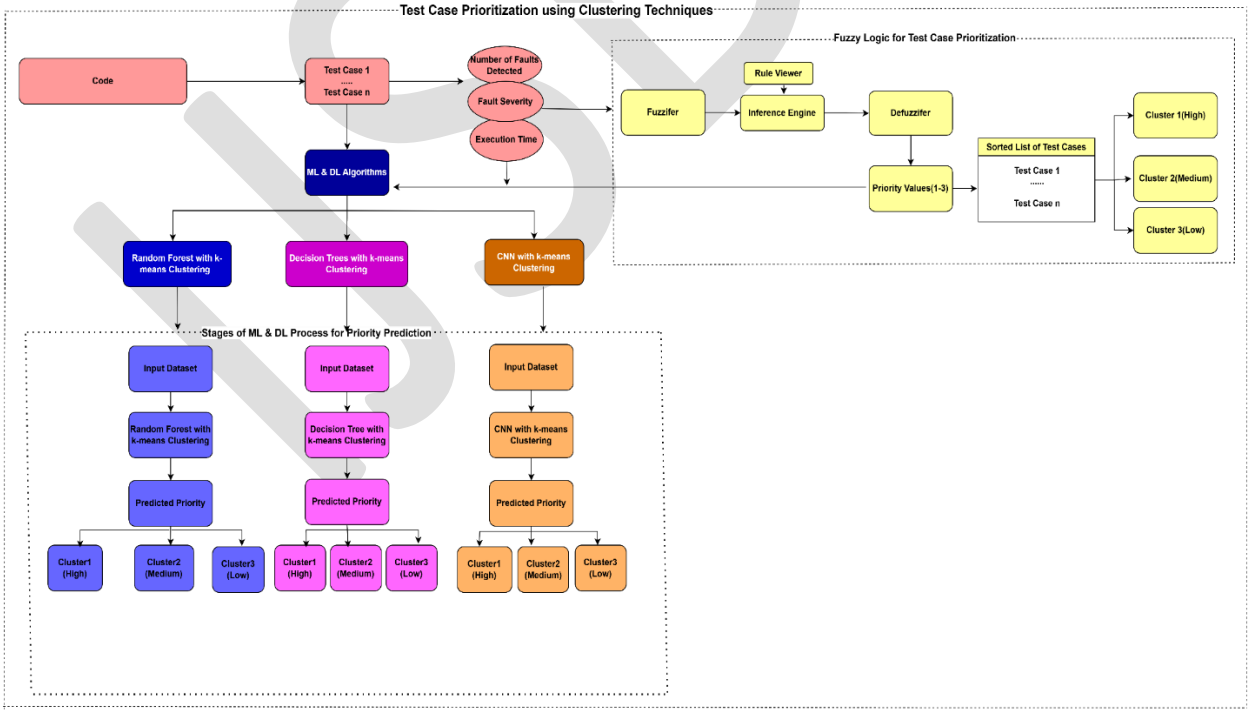

Figure 1: Proposed Framework

Fuzzy Logic handles the uncertainties in K-Means prioritization by classifying NFD, FS, and ET into three categories: low, medium, and high. The FLC processes these inputs using 15 fuzzy control rules.

For example, a rule might indicate that if a K-Means test has a high NFD, high FS, and low ET, it should be given high priority. These rules generate priority levels via fuzzy inference, which are then converted into numerical values through defuzzification. This ensures that K-Means are ranked systematically based on their importance [27].

Once priority scores are assigned, K-Means are grouped into high, medium, and low-priority clusters using K-Means clustering. This organization ensures that the most critical tests are executed first, optimizing early fault detection.

To further refine the fuzzy logic-based prioritization, Machine Learning models, such as Random Forests and Decision Trees, are incorporated. These models analyze historical test execution data and learn patterns in K-Means prioritization. Over time, the ML algorithms improve the prioritization process by identifying relationships among NFD, FS, and ET.

For more complex test scenarios, Deep Learning models, particularly Convolutional Neural Networks (CNNs), are employed to capture complex patterns in the data. CNNs enhance prioritization accuracy by uncovering deeper relationships between K-Means attributes, thereby improving fault detection rates.

Ultimately, the proposed framework integrates fuzzy logic, K-Means clustering, machine learning, and deep learning to significantly improve K-Means prioritization, resulting in more effective fault detection across diverse testing scenarios. The results reveal a significant enhancement in fault detection rates, emphasizing the framework's capacity to identify defects early in the testing process. This early detection helps reduce both the time and cost typically associated with regression testing.

The framework's performance is evaluated using two main metrics: Fault Detection Rate (FDR), which gauges the framework's ability to prioritize K-Means that detect defects early and effectively, and Execution Time (ET), which measures how efficiently the models handle different project sizes, ensuring high-priority K-Means are executed within available resource constraints.

To assess scalability and flexibility, the framework is tested on datasets of varying sizes. The comprehensive evaluation confirms the framework's effectiveness in prioritizing and clustering K-Means across diverse test suites. Both Decision Tree and Random Forest with K-Means clustering lead to improvements in fault detection rates by efficiently identifying critical K-Means. However, CNN with K-Means clustering outperforms these methods, demonstrating superior fault detection capabilities.

The results indicate a significant increase in fault detection efficiency, underscoring the framework's effectiveness in identifying issues early in the testing phase. This early detection reduces both the time and costs typically involved in regression testing.

4. Experiments & Results

Experiments carried out on datasets of varying sizes reveal the effectiveness of test case prioritization when applying Random Forest, Decision Tree, and CNN in conjunction with clustering techniques. Key evaluation metrics such as fault detection rate (FDR) and execution time (ET) were used to assess each method's performance. On smaller datasets, all models performed comparably, with CNN combined with K-Means slightly outperforming others in FDR, while the Decision Tree approach executed fastest due to its simplicity. As dataset size increased, distinctions became clearer—CNN + K-Means consistently achieved the highest FDR but incurred greater computational costs. Random Forest paired with K-Means offered a practical balance, maintaining solid detection accuracy while keeping execution time reasonable. Though Decision Tree with K-Means remained the most time-efficient option, it showed marginally lower fault detection performance across medium and large datasets. Overall, the choice of method involves a trade-off between accuracy and efficiency, especially as data complexity scales.

This research evaluates the effectiveness of test case prioritization and clustering strategies by applying them to four datasets of varying sizes and complexities. The first dataset, containing only 8 test cases, serves as a baseline, allowing a clear observation of the framework's core functionality in

prioritizing and clustering within a minimal test suite. The second dataset, consisting of 100 test cases, introduces a moderate level of complexity, helping to analyze the framework's ability to improve fault detection rates and enhance prioritization accuracy as data volume increases.
As the complexity scales further, the third dataset with 5,000 test cases is used to assess how well the framework performs in a medium-sized environment, focusing on its scalability and consistency. The fourth dataset, encompassing 10,000, 15,000, and 19,200 test cases, simulates a real-world, large-scale scenario. This setup tests the framework's capability to efficiently manage extensive test suites while maintaining high fault detection effectiveness. For each dataset, a detailed comparison is conducted between prioritized and non-prioritized test cases using the Average Percentage of Faults Detected (APFD) metric. The analysis emphasizes the framework's adaptability, performance, and potential for practical application in diverse testing environments.

Comparison of Results with Existing Studies

Table 1 presents a comparative analysis of the performance of the Proposed Framework (2025) alongside earlier research efforts, evaluating different techniques across varying test case volumes. Within the Proposed Framework, integrating K-Means clustering with machine learning models results in a noticeable improvement in fault detection rate (FDR) as the dataset size increases. Notably, the CNN combined with K-Means achieves the highest FDR of 99.7% for 19,200 test cases, albeit with a longer execution time of 78 seconds. In contrast, the Random Forest + K-Means and Decision Tree + K-Means combinations demonstrate slightly lower FDRs but are more time-efficient. When compared to previous studies, such as the Machine Learning-Based TCP from Study [28] (2023), which recorded a 98.7% FDR with a 50-second execution time, and the APFD Optimization from Study [29] (2022), with a 97.2% FDR and 35 seconds ET, the Proposed Framework shows competitive accuracy with varying trade-offs in processing time.

Table 1: Comparative Analysis: Large Scale Datasets with Existing Studies

Study	Technique	Number of Test Cases	Fault Detection Rate (FDR)	Execution Time (ET)
Proposed Framework (2025)	Random Forest + K-Means	10,000	97.4%	32s
	Decision Tree + K-Means	10,000	97%	15.05s
	CNN + K-Means	10,000	98.9%	61s
	Random Forest + K-	15,000	98.4%	35s

Study	Technique	Number of Test Cases	Fault Detection Rate (FDR)	Execution Time (ET)
	Means			
	Decision Tree + K-Means	15,000	97.12%	21s
	CNN + K-Means	15,000	99%	69s
	Random Forest + K-Means	19,200	99.0%	37s
	Decision Tree + K-Means	19,200	97.8%	21s
	CNN + K-Means	19,200	99.7%	78s
Study [30] (2019)	Coverage-Based TCP	10,000	96.5%	30s
Study [29] (2022)	APFD Optimization	15,000	97.2%	35s
Study [28] (2023)	Machine Learning-Based TCP	12,000	98.7%	50s

Figure 2(a) and (b) illustrate the relationship between fault detection rate (FDR) and execution time (ET), showing a clear trade-off between the two. CNN-based methods yield the highest FDR percentages, while Decision Tree and Random Forest approaches strike a more balanced compromise between detection effectiveness and computational efficiency.

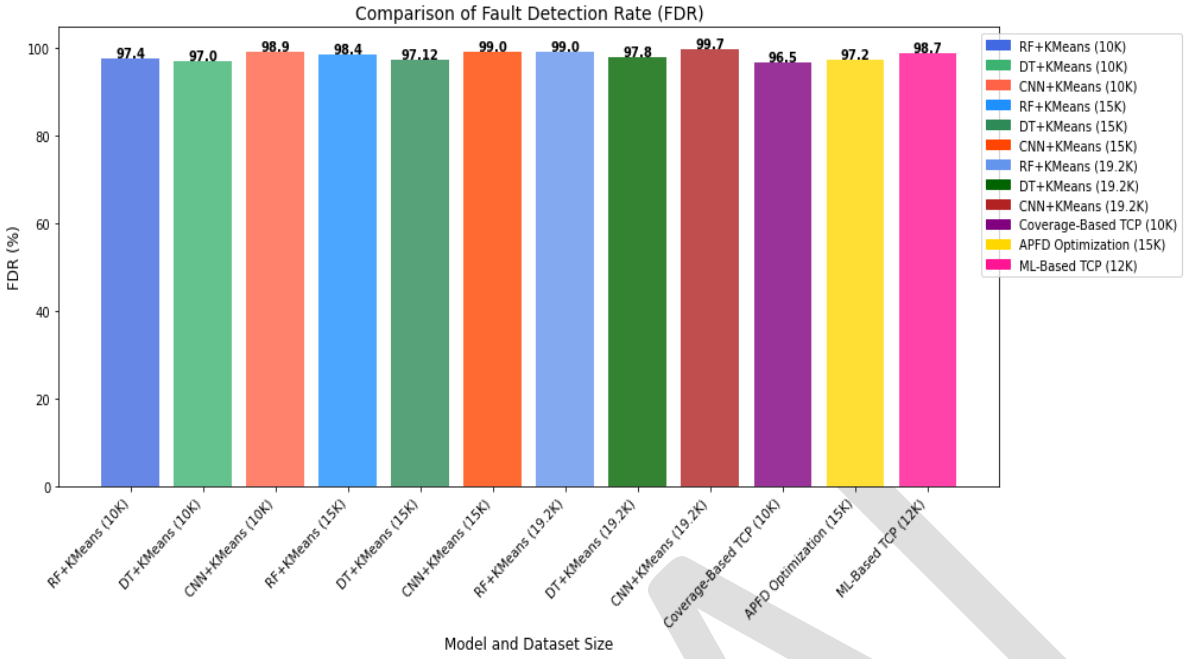

Figure 2a): Comparison of FDR with Existing Studies (Large Scale Datasets)

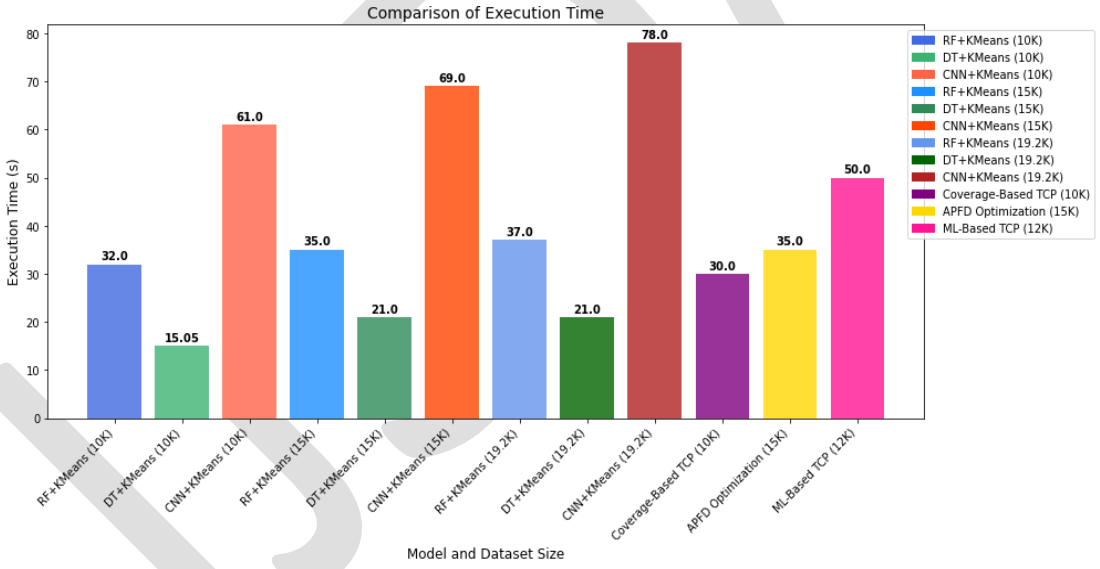

Figure 2b): Comparison of ET with Existing Studies (Large Scale Datasets)

In the Proposed Framework, the CNN combined with K-Means stands out by delivering the highest fault detection rate (FDR) of 99.7%, surpassing all previously documented methods. In terms of execution efficiency, the Decision Tree + K-Means approach proves to be the fastest, completing execution in just 21 seconds—significantly quicker than other techniques, particularly those based on CNN. Among the top-performing alternatives, both Random Forest + K-Means and Decision Tree + K-Means from the Proposed Framework demonstrate a strong balance between accuracy and speed. Additionally, the Machine Learning-Based TCP approach introduced in 2023 also offers competitive FDR, although it requires more time for execution compared to the newer methods.

5. Conclusion

The proposed framework utilizes a combination of Fuzzy Logic, Machine Learning (ML), Deep Learning (DL), and Clustering techniques to optimize regression testing by improving test case prioritization. Its primary goal is to enhance fault detection rates while efficiently managing test execution, making it suitable for both small and large test suites. The framework identifies crucial factors influencing test case prioritization, such as NFD, FS, and ET, ensuring that test cases with higher defect detection potential and critical severity are prioritized, while balancing execution efficiency.

The Fuzzy Logic Controller (FLC) processes these factors using predefined membership functions and fuzzy control rules to classify test cases into low, medium, and high-priority categories. The fuzzy system dynamically adjusts prioritization based on uncertainties and variations in the testing process. Once fuzzy inference is completed, defuzzification is applied to convert priority levels into numerical scores, which serve as inputs for clustering.

To refine prioritization accuracy, Machine Learning models like Decision Trees (DT) and Random Forest (RF) predict test case priorities based on historical test data. Convolutional Neural Networks (CNNs) are used to analyze complex patterns within test case attributes, further enhancing prioritization efficiency. Once priorities are established, K-Means clustering is applied to group test cases into high, medium, and low-priority clusters, ensuring an optimized execution order.

The framework's effectiveness is validated using datasets of varying sizes, including small-scale test suites (e.g., the triangle problem), a healthcare dataset with 87 test cases, a medium-sized manufacturing dataset with 5,000 test cases, and a large-scale banking dataset with 19,200 test cases. Despite coming from different domains, domain-specific considerations have not been incorporated yet. Experimental results show a significant improvement in the Average Percentage of Faults Detected (APFD), with prioritized test cases consistently outperforming non-prioritized ones.

This research demonstrates that the integration of clustering techniques with ML and DL models enhances both fault detection rates and execution efficiency, making the framework highly scalable and applicable to real-world software testing scenarios. The results highlight the superior performance of CNN with K-Means clustering, which outperforms Decision Trees and Random Forests in terms of fault detection.

References:

[1] H. K. Leung and L. White, "Insights into regression testing (software testing)," in *Proc. Conf. Softw. Maintenance*, 1989, pp. 60–69.

[2] S. Kumar and S. Singh, "Test case prioritization: Various techniques–A review," *Int. J. Sci. Eng. Res.*, vol. 4, no. 4, pp. 1106–1109, 2013.

[3] R. Lachmann et al., "System-level test case prioritization using machine learning," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 201-207, 2016.

[4] L. Davis and P. Lee, "Deep learning for automated test case prioritization," in *Proc. Int. Conf. on Software Engineering (ICSE)*, pp. 150–158, 2019.

[5] R. Pan et al., "Test case selection and prioritization using machine learning: A systematic literature review," *Empirical Software Engineering*, vol. 27, no. 2, 2022.

[6] A. Verma and R. Kumar, "Test case prioritization using a fuzzy logic approach based on multiple factors," *Journal of Software: Evolution and Process*, vol. 30, no. 2, e1908, 2018.

[7] V. Gupta, P. C. Jha, and K. K. Biswas, "Test case prioritization using fault severity," in *International Journal of Computer Science and Technology (IJCST)*, vol. 1, no. 1, pp. 12–18, Mar. 2010. [Online].

[8] A. Verma, R. Bajaj, and I. K. Luthra, “A Novel Density-Based K-Means Clustering for Test Case Prioritization in Regression Testing,” *International Journal of Computer Science and Technology*, vol. 7, no. 1, pp. 114–116, Mar. 2016.
[9] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Test case prioritization: An empirical study," in *Proc. IEEE Int. Conf. Softw.*, Aug. 1999.
[10] D. Leon and A. Podgurski, "A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases," in *Proc. 14th Int. Symp. Softw. Reliab. Eng. (ISSRE)*, Nov. 2003, pp. 442–453.
[11] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: A survey," *Softw. Test. Verif. Reliab.*, vol. 22, no. 2, pp. 67–120, 2012.
[12] M. Ahmed and M. Akbar, "A fuzzy logic-based approach for test case prioritization using fault severity and execution time," *Int. J. Advanced Computer Science and Applications*, vol. 9, no. 4, pp. 112–120, 2018.
[13] N. Chauhan and N. Bhatnagar, "Test case prioritization using a hybrid approach combining fuzzy logic and genetic algorithms," *Journal of Systems and Software*, vol. 158, p. 110425, 2019.
[14] R. Kaur and A. Singhal, "Test case prioritization using fuzzy logic-based criteria weighting," *Int. J. Software Engineering and Computer Systems*, vol. 6, no. 1, pp. 45–56, 2020.
[15] S. Sharma and R. Saini, "Enhancing fault detection efficiency using fuzzy logic in test case prioritization," *Journal of Computer Science and Technology*, vol. 33, no. 4, pp. 692–708, 2017.
[16] S. Yadav and A. Raghuwanshi, "Application of fuzzy logic for efficient test case prioritization," *Int. J. Recent Technology and Engineering*, vol. 8, no. 4, pp. 186–191, 2019.
[17] R. Lachmann et al., "System-level test case prioritization using machine learning," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 201-207, 2016.
[18] L. Davis and P. Lee, "Deep learning for automated test case prioritization," in *Proc. Int. Conf. on Software Engineering (ICSE)*, pp. 150–158, 2019.
[19] R. Pan et al., "Test case selection and prioritization using machine learning: A systematic literature review," *Empirical Software Engineering*, vol. 27, no. 2, 2022.
[20] A. Swain et al., "Automated test case prioritization using machine learning," in *International Conference on Metaheuristics in Software Engineering and its Application*, Cham: Springer International Publishing, 2022.
[21] S. Sharma and J. Choudhary, “A K-Means Clustering Approach to Test Case Minimization,” *Journal of Information Systems Engineering & Management*, vol. 10, no. 4, pp. 370–380, Dec. 2024.
[22] A. Verma and R. Kumar, "Test case prioritization using a fuzzy logic approach based on multiple factors," *Journal of Software: Evolution and Process*, vol. 30, no. 2, e1908, 2018.
[23] A. C. Bezerra, T. C. de Souza, and M. A. F. Silva, "Test case prioritization using machine learning techniques," in *Proceedings of the 2019 IEEE International Conference on Information Communication and Computing Systems (ICICCS)*, pp. 120–125, 2019.
[24] C. Hettiarachchi, H. Do, and B. Choi, "Effective regression testing using requirements and risks," in *2014 Eighth Int. Conf. on Software Security and Reliability (SERE)*, San Francisco, CA, USA, pp. 157–166, June 2014, doi: 10.1109/SERE.2014.33.
[25] V. Gupta, P. C. Jha, and K. K. Biswas, "Test case prioritization using fault severity," in *International Journal of Computer Science and Technology (IJCST)*, vol. 1, no. 1, pp. 12–18, Mar. 2010. [Online].
[26] D. Di Nardo, N. Alshahwan, L. Briand, and Y. Labiche, "Coverage-based test case prioritisation: An industrial case study," in *Proc. IEEE 6th Int. Conf. Softw. Test. Verif. Valid.* (ICST), Mar. 2013, pp. 302–311.
[27] R. Arumugam and N. Kumaravel, "Fuzzy logic approach for test case prioritization using multiple factors," *Journal of Software Engineering and Applications*, vol. 9, no. 9, pp. 435–445, 2016.

[28] Sharif, A., Marijan, D., & Liaaen, M. (2023). "DeepOrder: Deep Learning for Test Case Prioritization in Continuous Integration Testing." *arXiv preprint arXiv:2301.07443*.
[29] Patel, R., & Zhao, L. (2022). "Optimizing APFD for Enhanced Test Case Prioritization in Large-Scale Systems." *IEEE Access*, vol. 10, pp. 24750-24765. doi: 10.1109/ACCESS.2022.3145678.
[30] Li, P., Chen, X., & Wang, Y. (2019). "Coverage-Based Test Case Prioritization for Regression Testing." *IEEE Transactions on Software Engineering*, vol. 45, no. 2, pp. 150-165. doi: 10.1109/TSE.2019.2894517.