Volume (1) No (6), 2025
**Journal of Global Research in Multidisciplinary Studies (JGRMS)**
**Review Paper/Research Paper**
**Available online at** https://saanvipublications.com/journals/index.php/jgrms/index

# Looking into How Machine Learning is Used for Regression Testing Within the Agile Software Development Context

Dr. Bal Krishna Sharma
*Professor*
*Department of Computer Sciences and Applications*
*Mandsaur University*
Mandsaur
bksharma7426@gmail.com

***Abstract*—Regression testing is essential for maintaining software reliability, especially in agile development environments marked by frequent code changes and iterative delivery cycles. Traditional regression testing methods often fall short in keeping up with the speed and complexity of modern development, leading to a growing need for smarter and more efficient solutions. In this context, Machine Learning (ML) has been developed as a transformational method, offering intelligent capabilities such as test case selection, fault prediction, and change impact analysis. These developments ensure that new code modifications do not adversely affect current functionality by automating and optimizing testing. This paper presents a comprehensive review of how ML is being applied to enhance regression testing in agile settings, highlighting current methodologies, categorizing state-of-the-art applications, and outlining the benefits in terms of accuracy, efficiency, and automation. The review aims to support researchers and practitioners in leveraging ML to develop more effective and Agile software development techniques for adaptive regression testing.**

***Keywords*—Regression Testing, Agile Software Development, Machine Learning, Test Case Prioritization, Fault Prediction, Change Impact Analysis, Software Testing Automation, Intelligent Testing, Software Reliability.**

## I. INTRODUCTION

The significance of systems testing has increased in unison with the rapid evolution of software development and the rising need for high-quality as well as effective systems for software. The testing procedure evaluates the system and guarantees that it functions as needed since it confirms the program's error-free integrity, dependability in carrying out its mandate, and proper functioning [1]. There are several testing techniques, but the most often used are black box, white box, and grey box testing. Black box testing focusses on how the system works rather than going deeply into its fundamental parts [2]. The white box test focusses on the internal operations of the system, operating protocols, internal organization, and implementation. Grey box testing, on the other hand, blends the black box and white box testing methodologies. This sort of testing involves the tester performing functional testing of the software while taking into consideration its internal structure, after becoming familiar with it.

Many teams choose agile software development because it is a customer-focused methodology that speeds up software delivery to market through brief sprints. Additionally, projects that use agile software development are transparent, which improves cross-functional teamwork and communication [3]. Teams' work environments have an impact on their development process, which may either accelerate or slow it down. A Systematic Mapping Study was carried out [4]. An SMS aids in identifying topics and deficiencies that may be covered in further studies. It also aids in summarizing and defining the present focus of the investigation. Organizational management of workflows, teams, online tools, procedures, and established work norms may be better understood via the methodical examination of the confluence of hybrid work as well as agile software development.

ML models can forecast problems, create test cases, and optimize regression testing by utilizing data from previous software projects. As a result, testing now requires less human labor, and accuracy and coverage have increased [5]. ML-driven testing allows for the exploration of more exhaustive test cases, providing better coverage and reducing the likelihood of bugs escaping into production The integration of ML into software testing has far-reaching implications [6], including improvements in cost efficiency, faster release cycles, and more reliable software products. However, the application of ML is not without challenges.

### *A. Structure of the Paper*

This document follows the following format: Section II covers regression testing in agile environments. Section III explores ML applications in regression testing. Section IV highlights ML tools and techniques. Section V presents related literature, and Section VI ends with important discoveries and future projects.

## II. REGRESSION TESTING IN AGILE ENVIRONMENTS

Every software increment that is being designed and tested has a short testing period, and as was already said, regression testing has a very short testing period as well. Regression testing is comparable to other testing in that it allows all team members to develop tests, since agile promotes teamwork on tests. Regression testing entails retesting both the updated code and a particular impacted area of the original code. Regression testing automation looks like the best course of action. Regression testing in agile development contexts is

finding the automated method difficult, nevertheless, especially because of time restrictions [7]. If these automation scripts are created and updated on a weekly or nightly basis, the time required will become a limiting issue. In agile development settings, regression testing should prioritize tests for the following cycle in order to facilitate timely regression testing. As it can construct test cases, they may prioritize them using the details of upcoming release plans, which are specified by the requirements for each release. It may have strategies for using the knowledge that may be available for next release cycles, which might improve the efficacy and efficiency of the regression testing procedure going forward.

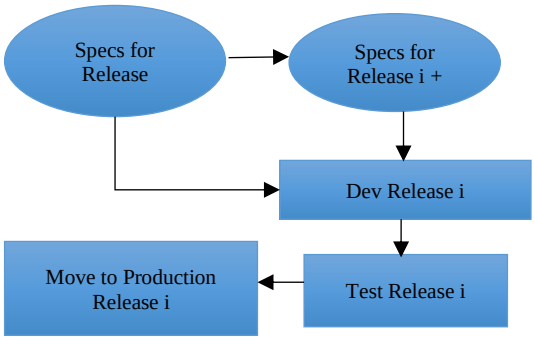


Fig. 1. Regression Testing in Agile Environment

The typical software release cycle for iteration "i", starting with specifications, followed by development, testing (see Figure 1), and production deployment. In agile configurations, regression testing is an essential part of the testing phase to ensure that new changes don't affect functionality that currently exists. Feedback from each cycle helps prepare specs for the next release, highlighting the iterative nature of the process.

### A. *Characteristics of Agile Projects*

Agile methods need less planning and divide tasks into digestible portions. The software development life cycle-compliant agile methodology is designed for short-term initiatives, including collaboration [8]. The software development life cycle includes the following phases: gathering requirements, designing, coding, analyzing, testing, and maintenance. To reduce software risks, customers as well as the management of the development team collaborate. This agile method's iterative nature permits modifications in response to customer satisfaction. Several iterations make it simple to introduce new features in an agile approach [9].

- **Iterative:** Agile software approaches prioritize client satisfaction through several iterations of a single need.
- **Modularity:** The whole system is divided into smaller, easier-to-manage pieces using an agile methodology. Modularity is essential to the software development process.
- **Time Boxing:** The iterative nature of the agile technique requires time limitations for each module and its accompanying cycle.
- **Parsimony:** Agile methods need parsimony to minimize risks and achieve goals with the fewest modules feasible.

### B. *Challenges of Regression Testing in Agile*

In an agile development setting, the regression testing team faces the following difficulties [10]:

- **Lack of Expertise:** As the project progresses, more specialists will be needed for more in-depth testing, such as integration and performance testing. To properly prepare test cases, the team should speak with experts inside the company to assess and collect requirements.
- **Frequent change in requirements:** Customers will occasionally request constant changes to their needs. This leads to unstable test cases that eventually launch the entire iteration.
- **Faster test suite growth:** As each sprint is released, the number of test suites increases as well. It becomes impossible to maintain such a huge test suite in a large-scale project. Frequently removing outdated test cases is necessary to guarantee proper upkeep.
- **Automation of test cases:** Test case automation shouldn't be a compulsory practice due to the increased maintenance needed. When and when automated test cases are required.
- **Insufficient communication among stakeholders:** Stakeholder communication must be smooth in order to fully comprehend the system change that is taking place. Respectful stakeholders ought to be informed about developments in other departments.

### C. *Advanced Regression Techniques*

The evolution of regression techniques has significantly transformed the landscape of predictive modelling, particularly within the realm of ML. Random forest regression is unique among the sophisticated techniques because it can efficiently handle non-linear correlations as well as interactions between variables [11]. This method improves accuracy and resilience by building numerous decision trees as well as aggregating their predictions using ensemble learning. In comparison, generalized linear models offer flexibility in modelling diverse types of response variables, effectively capturing underlying patterns in data. The application of these regression models extends beyond theoretical exploration; they play a crucial role in practical scenarios, such as optimizing vehicle routing problems, where objective values can be accurately predicted from input data without relying on traditional optimization techniques. Furthermore, studies demonstrate that advanced regression techniques also enhance fields like fire detection systems, achieving remarkable accuracy rates, as shown by the high performance of logistic regression at 99%. By harnessing these methodologies, researchers can robustly address complex real-world challenges and contribute to more effective decision-making processes.

## III. Machine Learning Applications In Regression Testing

Regression testing may now be improved with the use of ML, especially within Agile software development environments that require rapid and continuous testing. ML techniques are used to automate and optimize various aspects of regression testing, making it more efficient, targeted, and scalable. One of the most common applications is test case prioritization, in which ML models rank test cases based on their potential to uncover mistakes by reviewing previous test execution data, code updates, and defect patterns [12]. This helps in executing the most critical tests first, saving time and resources. Fault prediction is another important use case, where teams may concentrate their testing efforts on high-risk regions by using ML methods to identify codebase modules

that are more likely to have flaws [13]. Similarly, test suite minimization and selection use clustering and classification techniques to eliminate redundant test cases while maintaining adequate coverage. Change impact analysis benefits from ML by predicting how code modifications affect other components, guiding testers on which areas need re-evaluation. These applications not only reduce manual effort but also adapt dynamically to changing codebases, enabling smarter regression testing aligned with Agile principles. Overall, ML brings intelligence and automation to regression testing, improving its accuracy, speed, and effectiveness.

### *A. Machine Learning in Software Testing*

Time and ML in software testing have both helped software testing reach new heights. It was created as a result of the usage of ML in the validation of new software as well as applications. ML is a method of analyzing prior data to gain a better understanding and make better decisions about current and future challenges [14]. It comprises mathematical algorithms that can be modified based on data understanding and offer the most accurate answer possible. Neural networks used in deep learning, a kind of AI, replicate how the human brain operates. Deep learning can also make use of genetic algorithms and rule-based systems. However, neural networks are commonly used as a technology in deep learning systems.

### *B. ML Techniques for Fault Prediction*

There are two primary classifications for ML techniques:

- **Supervised learning:** In supervised learning, both the response variables and the predictors are provided. They provide many supervised learning algorithms, including Naïve Bayes, Decision Trees, and Random Forest.
- **Unsupervised learning:** In general, the unsupervised learning strategy is applied when no defect data is available. Here, the algorithm uncovers a pattern or hidden structure in the unlabeled data. If fault prediction requires defects to be predicted at several levels, clustering will be the most effective method.

### *C. Test Case Prioritization*

The key publications on test case prioritization as well as their historical significance. Eighty percent of the testing method is regression testing, which is essential for comparing software to revised requirements and quickly resolving defects. But because of its frequency, limited resources, and time-consuming nature, it poses difficulties [15]. Testing for regression in real-time embedded systems is very challenging due to the simulation environment's strict time constraints and the supervision of several efforts, including Retest-All, Regression Test Selection (RTS), Test Case Prioritization (TCP), and Test Suite Minimization (TSM). TCP is a method in regression testing that optimizes the order in which test cases are performed. Finding and running the most important test cases first is TCP's main goal, which enables the early identification of any software problems. Test cases that demonstrate at least one issue are chosen using a safe selection technique to achieve safe regression test selection, even if this does not always provide safety due to various circumstances. On the other hand, certain test cases are eliminated due to unsafe test case selection techniques. TCP highlights that during testing, test cases with a higher priority should be carried out faster. The time and money-efficient offline TCP method is not regarded as an expense.

### *D. Application of Regression Testing Techniques*

The worry that some potentially useful technology is being developed in academics but isn't always used in business. The term "academia-industry technology transfer gap" refers to the discrepancy between the cutting-edge methods that are suggested in academia and those that are actually employed in real-world software [16]. Raising awareness of the gap by voicing concerns about IR&A of RT techniques is critical, but it won't solve the challenge of actually implementing these approaches on its own. This section aims to determine if and to what degree techniques developed in the academic community have been applied to real-world software development. It expects that their live repository solution will identify these studies in the future and expand the results provided here, as previously said, there may be others that have been implemented but did not undergo their assessment because they were not specifically prompted by IR&A.

## IV. Machine Learning Tools And Techniques For Regression Testing

ML has introduced a new dimension to regression testing by enabling predictive, intelligent, and automated decision-making processes [17]. To prioritize test cases, analyze code changes, and find areas prone to errors, several ML methods are utilized, including SVM, decision trees, neural networks, as well as clustering approaches [18]. These models are trained using data collected in the past, including metrics for code complexity, defect logs, and change history to make informed predictions about where regressions are likely to occur. Effective use of these algorithms requires proper feature engineering transforming raw software metrics into meaningful input for ML models.

### *A. Open Source in Machine Learning*

Provides a high-level overview of open-source software and how it relates to other areas of science, notably ML. Avoiding to address any unfavorable viewpoints on open source might lead the reader to believe that they are excessively optimistic about its benefits [19]. The fact is that gathering hard information on the topic of proprietary systems vs open source software is incredibly difficult. They argue, on moral, ethical, and social grounds, that open source should be the primary software publication choice for ML research, and they point out the numerous benefits of open-source software development. The open scientific method also promotes the sharing of data as well as resources, which has several benefits. This article highlights the benefits of open-source software for ML research, which satisfies both scientific and software-related objectives.

### *B. Analysis Based on Machine Learning for Behavioral Distinction*

The researchers sought to see if they could distinguish the difference between sadness and anxiety by searching for a unique pattern of biased responsiveness to emotional stimuli in both conditions [20]. The test battery was devised to assist us in achieving their aim. The battery focuses on four types of biases: attention, memory, self-interpretation, and expectation biases. Even in the presence of large scoring noise and instrumentals, these methods enable the detection of complex nonlinear high-dimensional particular interactions that may impact output predictions. ML techniques based on decision tree methods were used especially. In order to categorize subjects into four groups, they were designed to be more sensitive: Low levels of anxiety and depression [LAD], low

 

levels of anxiety and depression [HD], low levels of depression and anxiety [HA], and high levels of anxiety and depression [HAD]. Each participant's self-report questionnaire responses were used to categorize their anxiety and depression levels, creating an asymptomatic profile. The ML model was trained using the behavioral tasks to infer each participant's symptomatic characteristics.

### *C. Integration with CI/CD Pipelines*

DevOps is a set of methods that mixes IT operations (Ops) with software development (Dev), with the purpose of minimizing the development lifecycle and continually providing high-quality software [21]. By promoting cooperation between development and operations teams, the DevOps concept dismantles conventional silos [22]. The agility and speed needed in contemporary software development are achieved through the use of DevOps techniques, such as CI/CD. Figure 2 illustrates how DevOps improves the theoretical foundation of CI/CD by highlighting automation, monitoring, and shared responsibilities—all of which are essential for successfully deploying CI/CD pipelines.

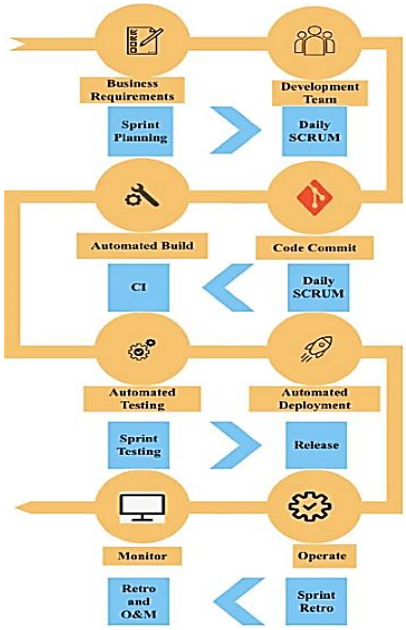


Fig. 2. Agile and DevOps Integration with CI/CD

An agile DevOps lifecycle starting from business requirements and sprint planning, followed by development with daily SCRUMs, code commit, automated build and testing, deployment, release, monitoring (see in Figure 2), and continuous improvement through retrospectives.

## V. Literature Review

This literature review section examines recent advancements, challenges, and research gaps in agile software development, focusing on regression testing, security integration, AI applications, and ML alignment within agile and continuous delivery contexts.

Das and Gary (2025) Methodically mapping research gaps as well as trends in regression analysis in agile environments, pinpointing areas that need more investigation to improve alignment with value-driven outcomes and agile processes. Method: 35 main studies were examined in a comprehensive mapping study. The study offered a thorough overview of the industry by classifying studies according to their methodology, agile frameworks, evaluation measures, and emphasis areas. Findings: The results place a high emphasis on test selection and prioritization, which reflects the necessity of agile workflows for optimal defect detection and execution efficiency. However, there is a lack of research in areas like cost analysis, test creation, and test minimization. The majority of current assessment metrics focus on technical results, ignoring agile-specific elements like iterative processes and the business effect of defect severity [23].

Valdés-Rodríguez et al. (2024) examines current approaches in the literature about safe software development in the context of small and medium-sized businesses utilizing agile software development techniques, with a 20.2% focus on SMEs in particular. The techniques that demonstrate how to effectively integrate safety features into the software development lifecycle are then analyzed and categorized. The results demonstrate how crucial it is to address security in an agile environment, since it remains a significant barrier in software development. Additionally, small businesses need to use effective tactics to ensure long-term profitability and application protection [24].

Selva-Mora and Quesada-López (2024) Agile software development is popular because it can quickly adapt to organizational demands, but it has trouble incorporating security procedures. Aligning security practices with agility is a tough task because, despite its effectiveness in providing prioritized functionality, non-functional requirements particularly security remain difficult to achieve. The Building Security in Maturity Model and the phases of the software development life cycle to classify 252 safety measures from 35 primary studies. Furthermore, it highlights 38 advantages, stressing safety awareness, implementation, as well as alignment with agility. It makes an effort to integrate security practices into Agile environments, emphasizing the value of empirical analysis and the need to ascertain the actual advantages of recommended security measures in real-world contexts. More agile that are software development [25].

Das (2024) Agile software development emphasizes rapid delivery of incremental features, raising concerns about software testing quality before release. Regression testing ensures that changes made to the code do not cause problems with features that have already been supplied. The challenge is to make sure that after stopping the delivery process, new code changes don't interfere with existing code. adapting existing regression test selection and prioritization algorithms to account for agile-specific process attributes such as time and value. The study tries to officially explain how regression testing is integrated within the framework of Agile Software Development. Additionally, the research goal is to develop regression test selection and Prioritization algorithms using ML techniques tailored to the characteristics of agile software development. Providing high-quality software within tight timeframes As well as revolutionizing the current approach to regression testing in agile software development are the goals [26].

Cabrero-Daniel (2023) Enhancing continuous delivery and integration by combining agile software development methodologies with artificial intelligence. An extensive literature survey as well as longitudinal meta-analysis of the articles that were obtained were used to investigate the role of AI and its possible applications in Agile software development. Critical issues, such the requirement for certain

 

socio-technical knowledge, were identified with the aid of the review. While better software development techniques might be facilitated by AI [27].

William et al. (2021) A software development environment that is agile. Poor task distribution might lead to client rejection, team members being demonized, as well as the project failing. Over the last decade, a large number of scholars have investigated various strategies for assigning labor in Distributed Agile. They used a variety of methodologies, including Bayesian networks and ontologies, which had not been used in the dispersed assignment of Agile software development jobs. To create and apply an ML-based job allocation technique in remote Agile software development, with the proposed model proving to be more accurate in work assignment [28]

Ranawana and Karunananda (2021) the alignment of ML production with normal software development methods is a common source of difficulty for software development teams. Depending on the production environment and real-time data adds another layer of complexity to the already difficult data gathering and preparation procedure. a standardized framework that streamlines software and ML engineering processes to make it easier to plan, build, and deploy ML applications. By incorporating, evaluating, and producing in real-time, the project and ML development risks may be greatly mitigated. Presenting a framework for the creation of ML applications, the MLASDLC brings together ideas from DevOps, MLOps, as well as normal software development life cycle techniques (SDLC) [29].

Table I summarizes studies on the issue of studying the use of machine learning for regression testing in agile software development systems. It covers the methodology, main findings, problems, and future prospects of this field

TABLE I. LITERATURE OF REVIEW ON LOOKING INTO HOW MACHINE LEARNING IS USED FOR REGRESSION TESTING IN AGILE SOFTWARE DEVELOPMENT ENVIRONMENTS

| Author | Study On | Approach | Key Findings | Challenges | Future Directions |
|---|---|---|---|---|---|
| Das and Gary (2025) | Exploring 35 Primary Studies in a Systematic Mapping Study (SMS) | Focus on test prioritization and selection- Emphasis on fault detection and execution efficiency- Categorized studies by metrics, frameworks | Under-explored areas: test generation, test minimization, and cost analysis- Narrow metric coverage | Explore test generation/minimization in agile- Incorporate cost and business impact metrics- Align more with agile principles | Regression testing in agile environments |
| Valdés-Rodríguez et al. (2024) | Secure software development in SMEs using Agile | Analysis of security practices in Agile SDLC | Emphasizes the need for security in Agile environments; identifies successful practices | SMEs face difficulties in integrating security effectively | Develop lightweight security frameworks tailored to SMEs |
| Selva-Mora and Quesada-López, (2024) | Security integration in Agile SDLC | Mapping of 252 security practices across BSIMM and SDLC stages | Highlights importance of aligning security with Agile; identifies 38 benefits | Lack of empirical validation; difficulty in alignment with agility | Evaluate effectiveness of practices in real-world Agile settings |
| Das (2024) | Regression testing in Agile using ML | Development of test selection and prioritization algorithms using ML | ML improves regression testing efficiency; better quality assurance in Agile | Handling time/value constraints and frequent code changes | Refine ML-based regression testing models for Agile environments |
| Cabrero-Daniel (2023) | AI integration in Agile development | Systematic literature review and meta-analysis | Identifies AI's potential in improving continuous delivery and integration | Requires socio-technical expertise and proper alignment | Enhance AI models for seamless Agile integration |
| William et al. (2021) | Job allocation in Distributed Agile | ML-based task assignment method | ML outperforms traditional methods (ontologies, Bayesian networks) | Lack of adoption in real Agile settings | Broaden ML usage in distributed Agile team management |
| Ranawana and Karunananda (2021) | Alignment of ML production with traditional software development processes | Proposed a standardized framework (MLASDLC) integrating DevOps, MLOps, and conventional SDLC practices | The framework simplifies planning, development, and deployment of ML applications while mitigating associated risks | Handling real-time data complexities and integrating ML workflows with existing software pipelines | Further refining MLASDLC to enhance real-time performance, scalability, and operational alignment |

## VI. CONCLUSION AND FUTURE WORK

Regression testing is a critical component of agile software development, and this section delves into how ML has revolutionized testing by making it more efficient and effective. Agile's iterative and fast-paced nature presents unique challenges for regression testing, such as time constraints, rapid requirement changes, and test suite maintenance. The integration of ML techniques such as test case prioritization, fault prediction, and change impact analysis has shown promising results in addressing these challenges by automating tasks, improving accuracy, and optimizing resource allocation. These advancements have not only minimized manual effort but have also made regression testing more intelligent and adaptive.

In the future, scientists should work on ML algorithms that are more reliable and use less reference data, making them more applicable in early stages of development or in data-scarce environments. Additionally, exploring hybrid testing frameworks that integrate human insight with ML-driven automation can help in overcoming current limitations. Further studies are also needed on adaptive learning models that evolve with the project and on creating standardized datasets and benchmarks to evaluate and compare ML-based regression testing approaches. Investigating domain-specific applications of ML in regression testing across industries could also provide valuable insights and improve generalizability.

## REFERENCES

[1] A. K. Polinati, “Devops and AI: Automating Software Delivery Pipelines for Continuous Integration and Deployment,” *Nanotechnol. Perceptions*, vol. 20, no. 4, 2024.

[2] S. I. Khaleel and R. Anan, “A review paper: optimal test cases for

 

regression testing using artificial intelligent techniques," *Int. J. Electr. Comput. Eng.*, vol. 13, no. 2, pp. 1803–1816, 2023, doi: 10.11591/ijece.v13i2.pp1803-1816.

[3] D. Khanna, E. L. Christensen, S. Gosu, X. Wang, and M. Paasivaara, "Hybrid work meets agile software development: A systematic mapping study," *Proc. - 2024 IEEE/ACM 17th Int. Conf. Coop. Hum. Asp. Softw. Eng. CHASE 2024*, pp. 57–67, 2024, doi: 10.1145/3641822.3641863.

[4] V. Prajapati, "Advances in Software Development Life Cycle Models : Trends and Innovations for Modern Applications," *J. Glob. Res. Electron. Commun.*, vol. 1, no. 4, pp. 1–6, 2025.

[5] A. Walvekar, "Leveraging Machine Learning for Software Testing and Quality Assessment," no. Ml, pp. 2–7, 2021.

[6] S. S. S. Neeli, "Key Challenges and Strategies in Managing Databases for Data Science and Machine Learning," *Int. J. Lead. Res. Publ.*, vol. 2, no. 3, p. 9, 2021.

[7] S. Faizullah and S. Almutairi, "Considerations for Regression Testing Process in Agile Development Environments," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 8, no. 1, p. 153, 2018, doi: 10.23956/ijarcsse.v8i1.565.

[8] S. R. Thota, S. Arora, and S. Gupta, "Al-Driven Automated Software Documentation Generation for Enhanced Development Productivity," in *2024 International Conference on Data Science and Network Security (ICDSNS)*, IEEE, Jul. 2024, pp. 1–7. doi: 10.1109/ICDSNS62112.2024.10691221.

[9] S. Sharma, D. Sarkar, and D. Gupta, "Agile Processes and Methodologies: A Conceptual Study.," *Int. J. Comput. Sci. Eng.*, vol. 4, no. 5, pp. 892–898, 2012.

[10] S. Chaudhary, "Regression Testing in Agile: Concepts, Strategies and Challenges," *Int. J. Res. Advent Technol.*, vol. 7, no. 5, pp. 418–421, 2019, doi: 10.32622/ijrat.752019218.

[11] N. Rawat, V. Somani, and A. K. Tripathi, "Machine Learning Approach for Regression Testing: A Case Study in Markov Chain Model," vol. 12, pp. 945–952, 2024.

[12] A. S and D. P. T, "Machine Learning for Automation Software Testing Challenges, Use Cases Advantages & Disadvantages," *Int. J. Innov. Sci. Res. Technol.*, vol. 5, no. 9, 2020.

[13] A. Goyal, "Scaling Agile Practices with Quantum Computing for Multi-Vendor Engineering Solutions in Global Markets," *Int. J. Curr. Eng. Technol.*, vol. 12, no. 06, Jun. 2022, doi: 10.14741/ijcet/v.12.6.10.

[14] P. Rasal, "Usage of Machine Learning Algorithms in Software Testing," *J. Orient. Inst.*, vol. 71, no. 4, 2022.

[15] J. Ahmad and S. Baharom, "A systematic literature review of the test case prioritization technique for sequence of events," *Int. J. Appl. Eng. Res.*, vol. 12, no. 7, pp. 1389–1395, 2017.

[16] R. Greca, B. Miranda, and A. Bertolino, "State of Practical Applicability of Regression Testing Research: A Live Systematic Literature Review," *ACM Comput. Surv.*, vol. 55, no. 13s, 2023, doi: 10.1145/3579851.

[17] Abhishek, A. Dhankar, and N. Gupta, "A Systematic Review of Techniques, Tools and Applications of Machine Learning," in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, IEEE, Feb. 2021, pp. 764–768. doi: 10.1109/ICICV50876.2021.9388637.

[18] A. Goyal, "Optimising Cloud-Based CI/CD Pipelines: Techniques for Rapid Software Deployment," *Tech. Int. J. Eng. Res.*, vol. 11, no. 11, pp. 896–904, 2024.

[19] S. Sonnenburg *et al.*, "The Need for Open Source Software in Machine Learning," *J. Mach. Learn. Res.*, vol. 8, no. 81, p. 2443−2466, 2007.

[20] M. R. Kishore, "A Review on Machine Learning Tools and Techniques," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 6, Jun. 2022, doi: 10.22214/ijraset.2022.44888.

[21] P. S. Emmanni, "Implementing CI/CD Pipelines for Enhanced Efficiency in IT Projects," *Int. J. Sci. Res.*, vol. 9, no. 9, pp. 1616–1619, Sep. 2020, doi: 10.21275/SR24402001528.

[22] G. Modalavalasa, "The Role of DevOps in Streamlining Software Delivery: Key Practices for Seamless CI/CD," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 1, no. 12, pp. 258–267, Jan. 2021, doi: 10.48175/IJARSCT-8978C.

[23] S. Das and K. Gary, "Regression Testing in Agile—A Systematic Mapping Study," *Software*, vol. 4, no. 2, p. 9, Apr. 2025, doi: 10.3390/software4020009.

[24] Y. Valdés-Rodríguez, J. Hochstetter-Diez, M. Diéguez-Rebolledo, A. Bustamante-Mora, and R. Cadena-Martínez, "Analysis of Strategies for the Integration of Security Practices in Agile Software Development: A Sustainable SME Approach," *IEEE Access*, vol. 12, pp. 35204–35230, 2024, doi: 10.1109/ACCESS.2024.3372385.

[25] A. Selva-Mora and C. Quesada-López, "Security Practices in Agile Software Development A Mapping Study," in *2024 IEEE/ACM International Workshop on Software-Intensive Business (IWSiB)*, 2024, pp. 56–63.

[26] S. Das, "Agile Regression Testing," in *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, IEEE, May 2024, pp. 457–459. doi: 10.1109/ICST60714.2024.00054.

[27] B. Cabrero-Daniel, "AI for Agile development: a Meta-Analysis," pp. 1–9, 2023, doi: 10.48550/arXiv.2305.08093.

[28] P. William, P. Kumar, G. S. Chhabra, and K. Vengatesan, "Task Allocation in Distributed Agile Software Development using Machine Learning Approach," in *Proceedings of IEEE International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications, CENTCON 2021*, 2021. doi: 10.1109/CENTCON52345.2021.9688114.

[29] R. Ranawana and A. S. Karunananda, "An Agile Software Development Life Cycle Model for Machine Learning Application Development," in *5th SLAAI - International Conference on Artificial Intelligence and 17th Annual Sessions, SLAAI-ICAI 2021*, 2021. doi: 10.1109/SLAAI-ICAI54477.2021.9664736.

 