Machine Learning-Based Anomaly Detection In CI/CD Pipelines for Reliable DevOps Automation

Mollety Dwaraka Venkata Naga Srikanth[1], Prathyusha Gali[2]
[1,2]*Independent Researcher, Affiliation Address: Hyderabad, Telangana, India*

***Abstract*—Continuous integration and continuous deployment (CI/CD) pipelines are vital tools in the DevOps tool box, issues like build failures, resource constraints, and deployment issues can lower system reliability and efficiency. This study proposes a novel framework called Anomaly Detection Framework using Machine Learning for enhancing the reliability of DevOps Automation. The steps of the methodology include data collection from the CI/CD Pipeline Failures Dataset for AIOps, data preprocessing, feature extraction, detection of anomalies, model training, and performance evaluation. The four machine learning models implemented and tested were Isolation Forest, Local Outlier Factor (LOF), Random Forest and Autoencoder. The experimental results showed that Random Forest model showed the highest accuracy, 96.3%, and precision, 95.6%, recall, 96.8%, and F1 score 96.2%. Further, the anomalous pipeline executions resulted in peaks of 90% and 85% CPU and memory utilization, respectively. The results show that machine learning algorithms can accurately identify anomalies, minimize operational risks, and markedly improve the reliability and efficiency of the CI/CD-driven DevOps automation.**

***Index Terms*—Machine Learning, Anomaly Detection, CI/CD Pipelines, DevOps Automation, Predictive Analytics, Software Reliability.**

I. INTRODUCTION

The use of DevOps in software development has revolutionized modern software development allowing for faster delivery cycles, continuous integration and continuous deployment (CI/CD). The CI/CD pipelines automate process of code integration, testing, building and deployment, thereby improve software quality and reduce time to market [1]. As software systems become more complex and distributed, CI/CD pipelines produce enormous amounts of operational data such as build logs, test results, deployments, performance data, infrastructure events, and more [2]. As these environments are dynamic, anomalies like build failures, deployment issues, security vulnerabilities, resource limitations, and configuration issues can occur. Such anomalies can lead to software deliveries being impacted, reduction of system reliability and higher operational cost [3].

Traditional rule-based monitoring methods fail when the anomalies they are looking for are unknown or changing over time as they depend on pre-defined thresholds and patterns. As CI/CD environments continuously evolve, manually maintaining such rules becomes challenging and inefficient. Recently, Machine Learning (ML) has shown great potential to overcome these drawbacks: Anomaly detection systems (ADS) based on ML have been proposed to detect anomalies. Through the analysis of pipeline historical data and real-time data stream, ML algorithms can learn normal pipeline behavior automatically, and detect deviations which may be the signs of pipeline failure or performance problems. This feature allows for proactive monitoring, quick incident response and increased reliability of DevOps automation [5, 6].

Advanced anomaly detection methods like supervised learning, unsupervised learning, deep learning, and ensemble approaches have been found to have great potential in identifying complex CI/CD anomalies [7, 8]. These techniques offer invaluable opportunities to uncover hidden patterns in vast amounts of data, eliminate false alarms and warn in advance of any critical failures. Moreover, incorporating anomaly detection models into DevOps pipelines enables predictive maintenance and smart decision-making, further boosting system resilience and operation efficiency [9, 10].

Continuous integration and delivery (CI/CD) pipelines are crucial for enhancing the reliability of DevOps automation. Continuous integration and delivery (CI/CD) pipelines are essential for making DevOps automation more reliable and robust. The study examines which types of models are appropriate to use for ML, and discusses the characteristics of the pipeline data before evaluating the performance of the anomaly detection models in terms of relevant metrics [11, 12]. The outcomes are expected to help to build intelligent monitoring solutions that can guarantee continuous, secure and reliable software delivery in the modern DevOps environment [13-15].

The rest of this paper is presented as follows. The related literature review on DevOps automation, CI/CD pipeline and anomaly detection techniques based on machine learning principles is discussed in Section 2. In Section 3, the proposed methodology is detailed such as data collection, data preprocessing, feature extraction and model building. The implementation details and experimental setup is discussed in section 4. The results and performance of the proposed models are given in Section 5. The conclusions, problems, and applications are discussed in section 6. The paper ends with Section 7 that discusses future directions for study.

The plans are carried out by accomplishing the objectives:

- To analyze common anomalies and failure patterns occurring in CI/CD pipelines within DevOps environments.
- To develop and implement machine learning-based models for automated anomaly detection using pipeline-generated data.
- To evaluate the effectiveness of different anomaly detection techniques in terms of accuracy, precision, recall, and false positive rate.
- To enhance the reliability, stability, and operational efficiency of DevOps automation through proactive anomaly detection.

II. LITERATURE REVIEW

New study has shed light on the critical role of anomaly detection and intelligent automation, using machine learning, for enhancing the reliability, efficiency, and performance of DevOps CI/CD pipelines. Gbemi et al. (2026) [16] introduced novel predictive analytics, reinforcement learning and anomaly detection-based machine learning optimized CI/CD pipeline. AI-driven DevOps automation proved to be highly effective with a reported 40% higher deployment rate, 35% lower build failures, and 30% better resource utilization. Retrieving data from the log files, Kumar et al. (2026) [17] compared build failure rates, deployment time, and throughput before and after the implementation of AI-based CI/CD optimization, concluding that the rates have dropped (from 21.4% to 8.7% build failures), while deployment time has reduced from 9.8 to 6.4 minutes and throughput has increased from 46 to 68 builds per day. Gradient Boosting model gave 93.5% accuracy to the prediction. However, Maverick et al. (2026) [18] came up with a hybrid machine learning framework that reduced the execution time of a pipeline by 42%, the percentage of failed pipelines detected during the process by 68%, and the resources consumed by 35%. Mustakallio et al. (2026) [19] studied AI-for-failure-detection tools in the context of CI/CD and determined that Dynatrace succeeded in detecting failures for 100% of the 47 pipeline runs. For Current Events, Vasquez et al. (2025) [20] designed a data-driven DevOps framework, which successfully reduced Mean Time to Detection (MTTD) by 45% and Mean Time to Recovery (MTTR) by 38%, enhancing the resilience and efficiency of the pipeline process. Segireddy et al. (2024) [21] studied anomaly detection in financial CI/CD pipelines and pointed out its importance in reducing operational, financial, and reputational risks as well as compliance and fault tolerance. Amougou et al. [22] have reported the case of reduced errors in deployments by 34% to 47% on use of AI for DevOps Automation, a decrease in pipeline execution time of 28% to 41%, and an increase in resource utilization of 20% along with an F1 score of 0.82 to 0.91. Fawzy et al. (2023) [23] presented the DevOps Anomaly Detection Framework (DADF) that achieved 96% accuracy, 87.5% precision, 100% recall, and a 93.3% F1-score in detecting anomalies from DevOps industrial projects. Kondaparthy et al. (2023) [24] created a machine learning-based model for congestion prediction of CI/CD pipelines with 94.6% accuracy, 0.943 F1-score and a ROC-AUC value of 0.9974 that can help detect and scale workloads in advance to manage the congestion.

III. RESEARCH METHODOLOGY

The general approach is to create a Machine Learning-based Anomaly Detection System to detect any anomalies in CI/CD pipelines and enhance the robustness of DevOps automation. Data Collection, Data Preprocessing, Feature Extraction, Anomaly Detection Model Development, Model Training, and Performance Evaluation are the 6 stages of the framework in figure 1.

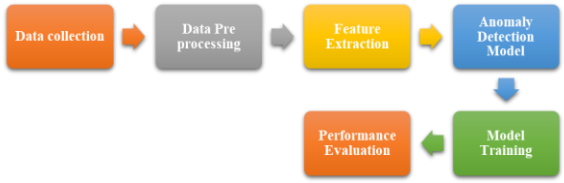

Figure 1: Framework of Methodology

3.1. Data collection

The CI/CD Pipeline Failures Dataset for AIOps is a specialized dataset for analysing failure and anomaly in Continuous Integration and Continuous Deployment (CI/CD) environments. It captures the logs of pipeline executions, build and deployment logs, failure information, execution durations, resource utilization data, and error data during the software delivery process. The dataset enables anomaly detection, failure prediction, and performance optimization study and analysis using machine learning techniques. It is especially well suited to train and evaluate models like Isolation Forest, Random Forest, Local Outlier Factor (LOF) and Autoencoders that enhance DevOps reliability and efficiency of automation.

3.2. Data Pre processing

Cleaning data is a crucial step in the proposed machine learning-based anomaly detection framework, and the raw data from CI/CD pipelines may be noisy, duplicate, noisy, and has missing values, which can affect the accuracy and reliability of the anomaly detection models. Preprocessing methods are used to enhance the data quality and for successful model training – missing value techniques, data normalization. The data set is filled with Mean Imputation method in which the average of a feature is used to fill the missing values. The mean value is the sum of all the values divided by the number of values:

$$\bar{X} = \frac{\sum_{i=0}^{n} X_i}{n} \qquad (1)$$

where $\bar{X}$ represents the mean value, Xi denotes an individual observation, and n is the total number of observations. Once handling (smoothing) the missing values occurs, all features get scaled between 0 and 1 by using Min-Max Normalization, which would not allow features with larger values to dominate the learning process. The normalization formula is:

$$X_{norms} = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (2)$$

where X is the original feature value, X_{min} is the minimum value, and X_{max} is the maximum value of the feature. By applying these preprocessing techniques, data that is more consistent and reliable, bias is reduced, and the performance of anomaly detection models in CI/CD pipelines are improved.

3.3. Feature Extraction

Feature Extraction is used to determine the most important features that aid in detecting anomalies in CI/CD pipelines. Build and deployment logs and execution history are analysed to retrieve relevant data, like error count, build time, failure frequency and so on, to expose the behaviour of pipelines. The total number of errors produced when running the pipeline is calculated as:

$$EC = \sum_{i=0}^{n} e_i \qquad (3)$$

where (e_i) represents each error occurrence. The build duration is computed by subtracting the time that elapsed from a build process starting to finish:

$$BD = T_{start} - T_{end} \qquad (4)$$

where (T_{start}) and (T_{end}) denote the build start and completion times, respectively. Failure frequency is the ratio of failed builds over all the builds, and is calculated as:

$$FF = \frac{N_f}{N_t} \qquad (5)$$

where (N_f) represents the number of failed builds and (N_t) denotes the total number of builds. The extracted features are critical inputs into the anomaly detecting model.

3.4. Anomaly Detection Model

- Isolation Forest

Isolation Forest is a machine learning algorithm that finds anomalies while randomly dividing data points. If there are some observations that do not conform with the rest, they would be identified and isolated quickly. The anomaly score is equal to:

$$s(x, n) = 2 - \frac{E(h(x))}{c(n)} \qquad (6)$$

where (E(h(x))) is the average path length, (c(n)) is the normalization factor, and (s(x,n)) represents the anomaly score. If (s(x,n)>0.5).

- Local Outlier Factor (LOF)

Local Outlier Factor (LOF) is a density-based anomaly detection method which detects anomalies by observing the difference in densities of the instance to those of its neighbors. Any points whose density is very different from nearby points are deemed to be anomalies. The value of LOF is computed as:

$$LOF(A) = \frac{\sum_{B \in N(A)} \frac{LRD(B)}{LRD(A)}}{|N(A)|} \quad (7)$$

where (LRD) denotes Local Reachability Density and (N(A)) represents the neighborhood of point (A). If (LOF>1), the observation is considered anomalous.

- Random Forest

Random Forest is a supervised machine learning algorithm which is a combination of multiple decision trees where they are used to classify normal and anomalous pipeline activities. It enhances prediction accuracy by combining the results of multiple trees and minimises overfitting. Gini Impurity is used for measuring a split:

$$Gini = 1 - \sum_{i=0}^{k} p_i^2 \quad (8)$$

where (pi) represents the probability of class (i) and (k) denotes the total number of classes.

- Autoencoders

The Autoencoder is a deep learning model for anomaly detection which learns the compressed representation of normal data. The anomalous instances give rise to greater reconstruction errors than normal observations during reconstruction. The reconstruction error is the total error between the reconstructed and actual values:

$$\text{RE} = \frac{1}{n}\sum_{i=0}^{n}(x_i - \hat{x}_i)^2 \quad (9)$$

where (xi) is the original input, ($\hat{x}_i$) is the reconstructed output, and (RE) is the reconstruction error. An observation is classified as anomalous when (RE>T), where (T) is the predefined threshold value.

3.5. Model Training

The dataset of the prepared CI/CD pipeline is split into two subsets for evaluation of the performance of the anomaly detection models. Approximately 80% of the data is used for training the machine learning algorithms, while the remaining 20% is used for testing and validation. The models are trained in the "normal" and “anomalous” pipeline behavior patterns. The 10-Fold Cross Validation technique is used to enhance the reliability of the model, its generalization capability, and to avoid overfitting. It is a technique which partitions the dataset multiple times into different training and validation sets and ensures that the models developed with this technique show accurate and consistent anomaly detection results with various data samples.

3.6. Performance Evaluation

Accuracy

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (10)$$

Precision

$$Precision = \frac{TP}{TP+FP} \quad (11)$$

Recall

$$Recall = \frac{TP}{TP+FN} \quad (12)$$

F1-Score

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (13)$$

IV. RESULT AND DISCUSSION

The CI/CD pipeline failure data were used to evaluate the proposed machine learning based Anomaly detection framework. Resource utilization metrics, anomaly detection accuracy, precision, and recall, F1 score, and training effectiveness were used to analyze the performance of the different models to identify the best model to improve DevOps automation.

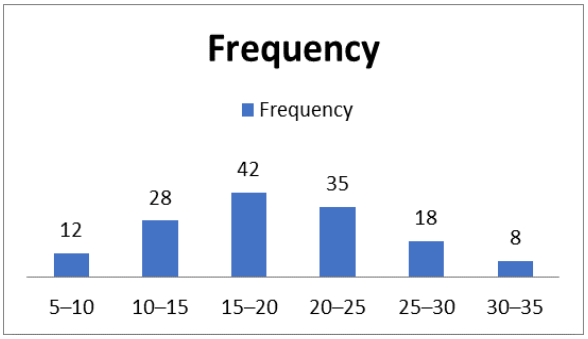

Figure 3: Build Duration Distribution of CI/CD Pipeline Executions

The build durations distribution in the CI/CD pipeline are shown in Figure 3. The 15–20-minute interval had the most amount of builds with 42, with the 20-25 minute interval coming in with 35 builds. Just 28 builds took 10-15 minutes and only 8 builds took 30-35 minutes. The results show that the majority of pipelines finish less than 15-25 minutes after beginning, while over 30 minutes might reflect some performance issues or strange pipeline activity that needs to be explored more.

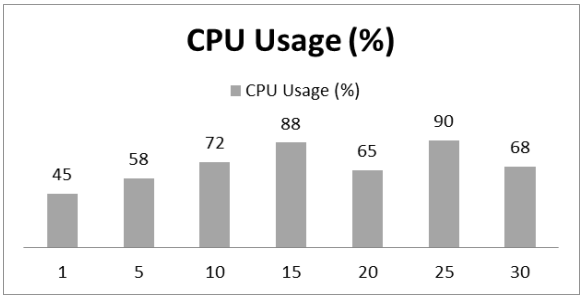

Figure 4: CPU Utilization during CI/CD Pipeline Execution

The CPU utilization of various pipeline runs is shown in figure 4. The CPU usage rose from 45% at Run 1 to a high of 90% at Run 25 and another high usage of 88% was at Run 15. Average CPU utilisation: ~69.4%. The increase in the significant spikes above 85% could suggest that there were times when the computational workload was high or that there was an anomaly in the pipeline's performance and reliability.

Figure 5 shows percentages of memory used for various runs of the pipeline. At Run 25, the memory used was at 85% of the allocated resources, the highest number of usages, while at Run 15, it used 81% of the resources.

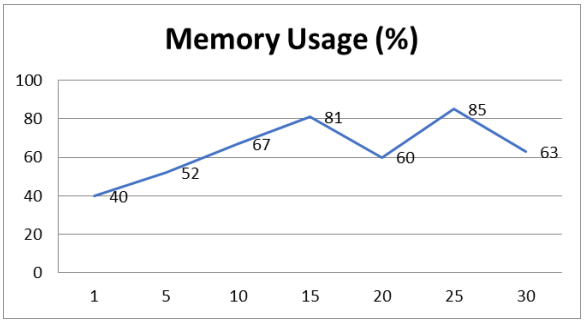

Figure 5: Memory Utilization During CI/CD Pipeline Execution

The average memory usage was around 64%. The changes in memory usage seen in runs 15 and 25 might be because of the memory intensive operations or some anomalous behavior which could impact CI/CD pipeline performance and system reliability.

Figure 6 shows the accuracy of various machine learning models for anomaly detection processes of CI/CD pipelines. The accuracy of the Random Forest model was 96.3%, higher than the accuracy of Autoencoder (94.5%) and Isolation Forest (91.2%). The accuracy of the LOF model was obtained as the lowest at 88.7%. As seen in these results, Random Forest is able to provide the best performance in terms of anomaly detection so it is the best model for reliable DevOps automation.

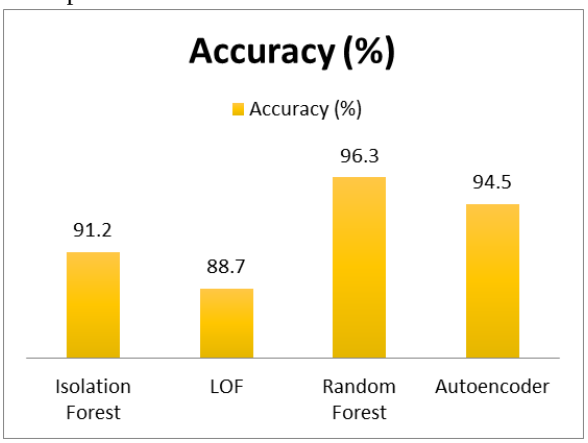

Figure 6: Accuracy Comparison of Anomaly Detection Models

A comparison of the Precision, Recall and F1-Score of the tested anomaly detection models is provided in Figure 7. The Random Forest model has the best Performance with a Precision of 95.6%, Recall of 96.8%, and F1-Score of 96.2%.

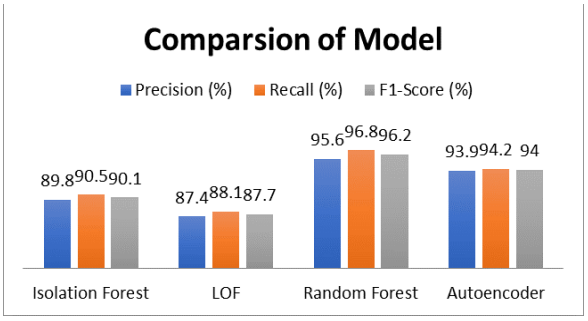

Figure 7: Precision, Recall, and F1-Score Comparison of Anomaly Detection Models

The Autoencoder model has the second-best Performance with Precision of 93.9%, Recall of 94.2%, and F1-Score of 94.0%. With 90.1%, Isolation Forest has the highest F1-Score while LOF has the lowest with 87.7%. The results in this section show that Random Forest gives the most accurate and reliable anomaly detection in CI/CD pipeline.

As shown in Figure 8, the training and testing accuracy have increased over the course of 20 epochs. From epoch 5 to 20, the accuracy rose to 78% and epoch 20 to 97% for training, while it rose from 75% to 95% for testing. The small gap between the training and test accuracy of 2%, after the last epoch, suggests a good model generalization and little overfitting. The results show that the proposed anomaly detection model has good detection and stability effects in CI/CD pipeline environments.

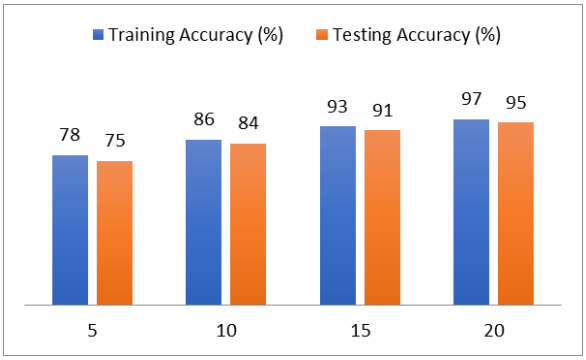

Figure 8: Training and Testing Accuracy across Epochs

V. CONCLUSION

This study introduced a framework based on Machine Learning for enhancing the reliability of CI/CD pipelines in DevOps. It employed data preprocessing, feature extraction, and machine learning models such as Isolation Forest, Local Outlier Factor, Random Forest, and Autoencoder for recognizing abnormal pipeline actions. Experimental analysis indicated that the build duration is mostly between 15 and 25 minutes and the maximum CPU and memory usages were found to be 90% and 85% respectively, during an anomalous build. The best results, in terms of accuracy, precision, recall and F1-score, were obtained by Random Forest model (96.3%, 95.6%, 96.8%, 96.2%, respectively). The Autoencoder also performed very well, with 94.5% accuracy. Moreover, the accuracy of the training and testing is raised to 97% and 95% respectively with good generalization of the model and without overfitting. The findings of these results validate the ability of machine learning-based Anomaly Detection to further improve the reliability of the CI/CD pipeline, minimize failure risk, and improve resource utilization and efficient DevOps automation.

REFERENCES

[1] Kolawole and A. Fakokunde, "Machine learning algorithms in DevOps: Optimizing Software Development and deployment workflows with precision," *International Journal of Research Publication and Reviews*, vol. 6, no. 1, p. 7421, 2025.

[2] J. Adebowale, "Adaptive DevOps automation with Generative AI: Enhancing CI/CD orchestration and predictive deployment in modern software delivery," Monograph/Preprint, 2025.

[3] A. Enemosah, "Enhancing DevOps efficiency through AI-driven predictive models for continuous integration and deployment pipelines," International Journal of Research Publication and Reviews, vol. 6, no. 1, pp. 871–887, 2025.

[4] S. R. Dileepkumar and J. Mathew, "Optimizing continuous integration and continuous deployment pipelines with machine learning: Enhancing performance and predicting failures," Advances in Science and Technology Research Journal, vol. 19, no. 3, 2025.

[5] H. E. Blackwood, "End-to-End Intelligent Automation Using AI and Machine Learning in DevOps Pipelines for Real-Time Analytics," International Journal of Multidisciplinary Research in Science, Engineering, Technology & Management, vol. 1, no. 1, pp. 25–32, 2025.

[6] S. Jain, "Integrating Artificial Intelligence with DevOps: Enhancing continuous delivery, automation, and predictive analytics for high-performance software engineering," World Journal of Advanced Research and Reviews, vol. 17, no. 3, pp. 1025–1043, 2023.

[7] P. Halvorsen, "Intelligent Software Testing and Continuous Delivery Frameworks for Financial Platforms with Machine Learning–Based Fraud Prevention," International Journal of Research

Publications in Engineering, Technology and Management (IJRPETM), vol. 6, no. 5, pp. 9755–9764, 2023.

[8] U. Bhatnagar and R. Mohana, "Diabetic Prediction Using DevOps CI, CD Pipeline," Working Paper/Report, 2023.

[9] S. Thalary and A. Katipelly, "Secure-by-Design Cloud Software Delivery: How DevOps and Software Teams Co-Own Security Outcomes," International Journal of Artificial Intelligence, Data Science, and Machine Learning, vol. 4, no. 1, pp. 131–140, 2023.

[10] B. Venkata, "AI-Powered Debugging: Exploring Machine Learning Techniques for Identifying and Resolving Software Errors," Working Paper/Report, 2023.

[11] P. Kale, "A Deep Learning-Based Platform Engineering Framework for Predictive CI/CD Pipeline Optimization and Developer Productivity Enhancement," International Journal of Artificial Intelligence, Data Science, and Machine Learning, vol. 5, no. 2, pp. 194–202, 2024.

[12] D. Takkalapally, "ShiftLeft-AI: Machine Learning Framework for Proactive Performance Assurance in CI/CD Pipelines," International Journal of Artificial Intelligence, Data Science, and Machine Learning, vol. 5, no. 4, pp. 285–296, 2024.

[13] I. Wilder, "Leveraging AI for Anomaly Detection and Performance Optimization in DevOps," Working Paper/Report, 2024.

[14] S. C. Vankayala, "Intelligent Failure Prediction in CI/CD Pipelines Using Machine Learning Models for Enterprise Quality Assurance," Working Paper/Report, 2022.

[15] M. V. Velumani and P. K. Muthukamatchi, "Cloud-Native Software Defect Prediction: Leveraging Machine Learning Models in Scalable CI/CD Pipelines," in Proceedings of the 2025 6th International Conference on Electronics and Sustainable Communication Systems (ICESC), 2025, pp. 1526–1531.

[16] A. Gbemi, "Intelligent CI/CD Pipeline Optimization Using Machine Learning-Based DevOps Automation," 2026, Available: SSRN 6179365.

[17] R. Kumar, A. Kumar, R. Kumar, S. Kumar, and D. N. D. Kollipara, "Artificial Intelligence–Based Optimization of CI/CD Pipelines for Reduced Build Failures and Deployment Time," Spectrum of Engineering Sciences, vol. 4, no. 3, pp. 1803–1812, 2026.

[18] A. Maverick, "Machine Learning–Based CI/CD Pipeline Optimization for Scalable Software Delivery," Working Paper/Report, 2026.

[19] T. Mustakallio, "Artificial intelligence for failure detection in CI/CD pipelines," Thesis/Report, 2026.

[20] E. Vasquez and J. O. Adebayo, "Data-Driven DevOps: Leveraging AI for Continuous Improvement in CI/CD Lifecycle Management," Working Paper/Report, 2025.

[21] A. R. Segireddy, "Machine Learning-Driven Anomaly Detection in CI/CD Pipelines for Financial Applications," Journal of Computational Analysis and Applications, vol. 33, no. 8, 2024.

[22] R. S. E. Amougou, "AI-driven DevOps: Leveraging machine learning for automated software delivery pipelines," Communication in Physical Sciences, vol. 9, no. 4, pp. 1010–1021, 2023.

[23] A. H. Fawzy, K. Wassif, and H. Moussa, "Framework for automatic detection of anomalies in DevOps," Journal of King Saud University - Computer and Information Sciences, vol. 35, no. 3, pp. 8–19, 2023.

[24] S. C. Kondaparthy, "DevOps Pipeline Bottlenecks Under Synthetic Load Spikes," 2023, Available: SSRN 6092146.

[25] M. Y. Abdullah, "CI/CD Pipeline Failure Logs Dataset for AIOps," Kaggle, 2026. [Online]. Available: https://www.kaggle.com/datasets/mirzayasirabdullah07/cicd-pipeline-failure-logs-dataset-for-aiops