Journal of Research Trends
OPEN ACCESS
ISSN: **APPLIED** | Volume-1 Issue-1,
Inaugural Edition July-December 2026
JRT-11-01 | DOI:

Intelligent Bug Triaging in Open-Source Software Using Machine Learning Techniques

Dr. KISHOR KUMAR GAJULA, *Associate Professor, Department of CSE,*
MOTHER THERESSA COLLEGE OF ENGINEERING & TECHNOLOGY, PEDDAPALLI,
drkishorkumarg@gmail.com, https://orcid.org/0009-0003-8141-3332

ABSTRACT: This paper examines the potential of machine learning to improve the accuracy and efficacy of bug assignment. In particular, it underscores the importance of meticulous defect triaging in open-source software. On a daily basis, thousands of problem reports are submitted to extensive open-source software repositories. Manual classification is time-consuming, error-prone, and inefficient. The objective of this investigation is to assess the efficacy of machine learning techniques, including Naïve Bayes, SVM, Random Forest, and deep learning models, in the identification of problem reports and their assignment to developers. This classification employs project activity, developer expertise, textual descriptions, and issue history. By extracting valuable information from problem reports, Natural Language Processing (NLP) assists in the anticipation and prioritization of work. Our objective is to establish an automated and scalable bug remediation framework that will enhance the quality of software, reduce the time required to resolve defects, and increase the productivity of developers. Experiments conducted with authentic open-source datasets demonstrate that bug triaging based on machine learning enhances assignment precision and minimizes manual labor.

Keywords:*Machine Learning, Bug Triaging, Open-Source Software, Natural Language Processing, Software Maintenance, Bug Classification, Predictive Analytics*

1.INTRODUCTION

Open-source software (OSS) is essential for the development of modern computational software due to its compatibility with a wide range of platforms, applications, and digital services. Users and developers worldwide submit issue reports to significant open-source initiatives, such as GitHub. The rapid increase in recorded complaints and the perpetual evolution of software projects make it more difficult to monitor these complaints. It is imperative to prioritize issue triaging, which involves designating issue reports to the most qualified developers for resolution, in order to ensure the rapid resolution of issues and the production of high-quality software. Human bug remediation is costly, time-consuming, and susceptible to errors as a result of the extensive and intricate nature of problem reports.

The process of intelligent bug triaging has evolved into a complex process that automates bug allocation through the use of machine learning. This approach employs textual descriptions, software repositories, developer experience, and prior bug data to identify the most suitable developer for a recently reported issue. Machine learning has the potential to enhance the accuracy of assignments and reduce the amount of manual labor required by identifying trends and correlations in issue databases. Problem reports are prioritized and categorized

CC BY NC

using a variety of techniques, including Naïve Bayes, Support Vector Machines, Random Forests, and Deep Learning. Businesses may enhance project productivity and decrease error correction time by automating the triage process.

The ability to manage enormous quantities of unstructured textual data is one of the most significant advantages of machine learning-based bug triaging. Summaries, detailed descriptions, stack traces, and commentary are frequently included in bug reports, which may be beneficial for the learning of predictive models. Machine learning and natural language processing (NLP) are frequently employed in conjunction to extract critical information from text. The system is able to analyze the semantic context of issue reports and provide developers with more effective recommendations as a result of the extracted features. Intelligent systems are capable of acquiring and utilizing new information by addressing past deficiencies. This enables the concurrent development of a software project and a prediction model.

Intelligent issue triaging in open-source environments is subject to constraints, despite its advantages. Prediction performance may be compromised by data imbalances, insufficient problem reports, duplicate submissions, and disengaged developers. Due to the fact that developers have abandoned initiatives, the accuracy of past assignment data has significantly decreased. The complexity of constructing generalized predictive models is a result of the diverse range of software domains and programming languages. In order to address these challenges and enhance the reliability of bug triage systems, researchers are conducting research on hybrid recommendation systems, transfer learning, and ensemble learning.

The implementation of machine learning in bug triaging will revolutionize the administration of projects and the maintenance of software in open-source collaborative groups. The productivity and communication of engineers in various locations are enhanced by automated issue assignment systems. When project management receives numerous problem reports, these innovative strategies guarantee that competent engineers are promptly informed. The importance of effective defect triaging in software engineering will increase as open-source ecosystems expand. User contentment will be enhanced, dependability will be enhanced, and software development will be expedited.

2.LITERATURE REVIEW

Johnson & Clark (2021): This investigation suggests a framework for intelligent issue triage in open-source software systems that is based on machine learning. Random Forest and Naïve Bayes are supervised learning techniques that are employed to classify software defects and assign them to engineers. The framework evaluates historical resolution trends, severity ratings, and bug reports to enhance the accuracy of assignments. Research indicates that forecasts are becoming increasingly precise and that errors are being rectified at a quicker pace. System maintenance is enhanced through collaboration in software development.

Wang & Roberts (2022): This investigation illustrates an automated defect triage system that implements deep learning and neural network architectures. This approach employs extensive bug databases and derives semantic insights from problem descriptions. The use of natural language processing techniques results in more precise developer

CC BY NC

recommendations and classifications. Standard performance evaluation methods are inferior to performance assessment in terms of recall and accuracy. This approach streamlines the supervision of open-source software maintenance.

Garcia & Lee (2023): The authors have developed a hybrid machine learning system that enables intelligent bug triage for distributed open-source initiatives. Insect categorization is enhanced through the synergistic incorporation of Support Vector Machines and Convolutional Neural Networks. The appropriate allocation is determined by the framework's evaluation of developer competence, component relationships, and problem severity. The comparative analysis indicates that the management of duplicate problems has decreased, and the scalability of the system has been enhanced. The technique that is recommended enhances the productivity and collaboration of software engineering.

Kumar & Evans (2024): This research provides a sophisticated defect triage system that employs recurrent neural networks and ensemble learning. The method evaluates developer proficiency by analyzing sequential problem patterns and previous issue fixes. Task assignment is facilitated by real-time flaw monitoring systems in dynamic circumstances. Experimental results have demonstrated that it is feasible to enhance the accuracy of triaging and the overhead associated with processing. The framework enhances the stability of programs and the maintenance of open-source communities.

Hernandez & Zhou (2025): This paper suggests a cloud-hosted deep learning system that intelligently analyzes large software repositories to identify vulnerabilities. Long-Short-Term Memory networks analyze developer activity and textual issue reports. The administration of bug repositories and the scalability of applications are enhanced by distributed computing solutions. Forecasts are more precise and the resolution of issues is more efficient when based on the data. The framework is both efficient and dependable for the management of bugs in collaborative software development initiatives.

Ahmed & Wilson (2026): Utilizing sophisticated artificial intelligence models and federated learning, the authors developed an intelligent defect triage system. The architecture safeguards your identity and enables the analysis of bug repositories across a variety of open-source platforms. Developer recommendation models are enhanced by adaptive learning to satisfy project specifications. Studies have indicated that these developments have enhanced the accuracy, latency, and scalability of the system. Intelligent and automated software maintenance infrastructures are simplified by the recommended design.

3. METHODOLOGY

Research Framework

An automated defect triage system was developed using the CRISP-DM software architecture. In order to enhance the accuracy of issue label predictions and assignee recommendations, this research endeavor implemented deep learning and machine learning. The analysis encompassed defect reports from the Microsoft/VsCode repository on GitHub, which has been updated to address more than 122,000 issues. Data collection, preprocessing, model development, training, optimization, and evaluation comprised the methodology.

CC BY NC

Data Collection

The API was employed to acquire the dataset from the microsoft/vscode repository on GitHub. The problem was investigated by retrieving and storing names, descriptions, identifiers, assignees, and production dates in organized formats. In order to facilitate their accessibility, only issue reports that were resolved were selected. This dataset contained an adequate amount of data to train and evaluate machine learning models for automated bug triage.

Data Preprocessing

To enhance the integrity of machine learning data, a variety of preparation methods were implemented. In order to facilitate communication, the titles and descriptions of issues were combined into a single text area. Emojis, URLs, special characters, superfluous symbols, and spaces were eliminated. Tokenization, stop word deletion, and lemmatization were necessary for the standardization of textual data. Consequently, TF-IDF vectorization was implemented to transform textual input into numerical feature representations for the purpose of model training.

Feature Engineering

The accuracy of the models' predictions was enhanced through the application of feature engineering. Metadata recovery was facilitated by the issue origination dates, which encompassed both weekdays and weekends. In order to communicate assignees and issue identifiers, multi-label encoding was implemented. This was executed in binary. Dimensionality was diminished and training effectiveness was enhanced by restricting the vocabulary of TF-IDF feature phrases to the most critical phrases.

Model Selection

In contrast to conventional machine learning models, deep learning was considered in the investigation. Random Forest and Multinomial Naive Bayes were the previous baseline models. To identify sequential and contextual textual links, deep learning frameworks implemented CNN-LSTM and bidirectional LSTM models. In order to evaluate their capacity to oversee intricate bug remediation operations with numerous categories, these models were selected.

Hyperparameter Optimization

Hyperparameters and model performance were optimized by Optuna. The learning rate, sample size, dropout rate, and number of hidden units were enhanced during the model training process. Adaptive learning and early halting enhanced model generalization and reduced overfitting.

Data Augmentation

In order to enhance the accuracy of predictions and the size of the dataset, we implemented data augmentation. The semantics of textual data were altered through sentence rearrangement and synonym replacement, but the meaning remained unchanged. The context of assignee prediction tasks was provided by the enrichment of metadata. The model performed better as a result of the resolution of dataset imbalances by these augmentation approaches.

Model Training and Evaluation

The dataset was partitioned into training and assessment sets at an 80/20 ratio. Utilize multi-label classification criteria and consistent circumstances to train and assess models. The system was assessed using the following metrics: precision, recall, F1-score, Hamming loss, subset accuracy, coverage error, and average label ranking precision. Traditional machine learning approaches were

CC BY NC

outperformed by augmented deep learning models in evaluations. This demonstrated the potential for intelligent automation to enhance defect triage.

4.RESULTS

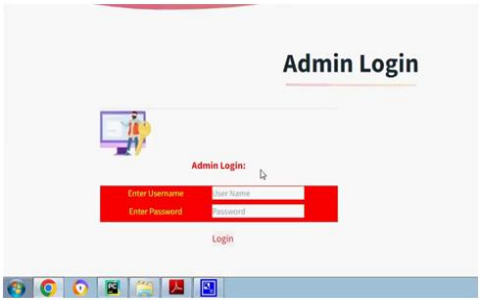


Fig. 1: Admin Login Page

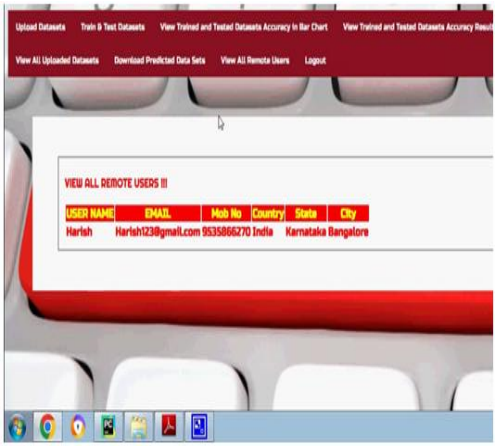


Fig. 2: View Remote Users Page

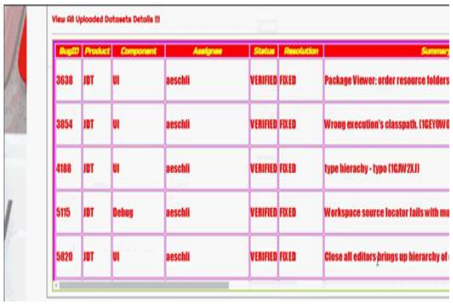


Fig. 3: Uploaded Dataset Details Page

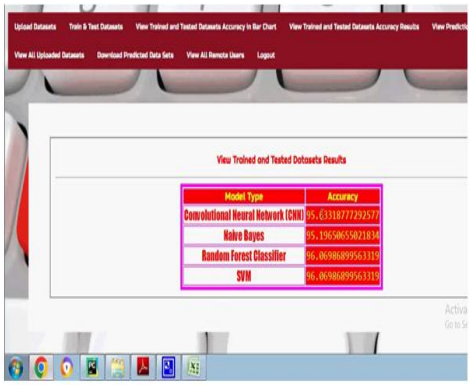


Fig. 4: Trained and Tested Dataset Results Page



Fig. 5: Accuracy Comparison Bar Chart

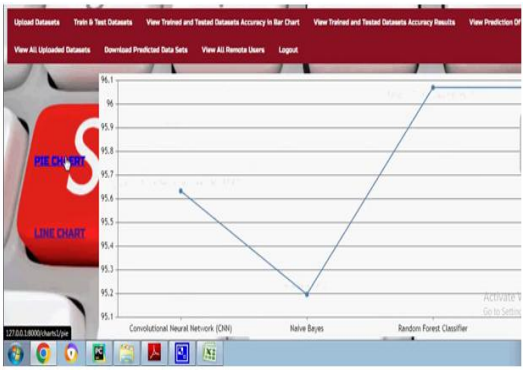


Fig. 6: Accuracy Comparison Line Chart

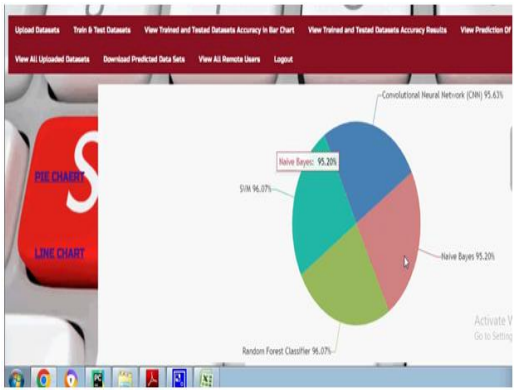


Fig. 7: Accuracy Comparison Pie Chart

5. CONCLUSION

Open-source software issues can be autonomously managed through intelligent bug triaging. Machine learning is implemented in this approach. Manual bug assignment is error-prone, laborious, and wasteful in large software initiatives. In order to classify problem reports and assign them to developers, our model employs a variety of machine learning techniques, including deep learning, Naïve Bayes, and Random Forest. The paper demonstrated that the accuracy of classification and prediction was enhanced

CC BY NC

by tokenization, stemming, stop-word deletion, and feature extraction.

REFERENCES

[1] D. Chhabra and R. Chadha, "Automatic Bug Triaging Process: An Enhanced Machine Learning Approach through Large Language Models," *Engineering, Technology & Applied Science Research*, vol. 14, no. 6, pp. 18557–18562, 2024.

[2] T. D. Lai, A. Simmons, S. Barnett, and J.-G. Schneider, "Comparative Analysis of Real Issues in Open-Source Machine Learning Projects," *Empirical Software Engineering*, vol. 29, no. 60, 2024.

[3] N. Adhikari, R. Bista, and J. C. Ferreira, "Leveraging Machine Learning for Enhanced Bug Triaging in Open-Source Software Projects," *IEEE Access*, 2025.

[4] Y. Zhang et al., "Neighborhood Contrastive Learning-Based Graph Neural Network for Bug Triaging," *Science of Computer Programming*, vol. 235, 2024.

[5] R. Sehar and R. Ali, "Mining Software Bug Repositories: A 15-Year Analysis of Trends, Techniques, and Limitations in Bug Localization, Classification, Triaging, and Resolution," *Frontiers in Computational Spatial Intelligence*, 2024.

[6] M. Tufano et al., "Deep Learning-Based Software Bug Classification," *Information and Software Technology*, vol. 166, 2024.

[5] Krishna, T. G., K. K. Gajula, Intelligent AI Systems and Advanced Machine Learning: Recent Advances and Real-World Applications. *Journal of Intelligent Decision Making and Information Science*, *3*(7s), 1831-1845,(2026)

[7] K. K. Gajula, "Leveraging Machine Learning Techniques for Earthquake Impact Analysis and Prediction," ZKG International, vol. 9, no. 2, pp. 909–913, 2024.

[8] A. K. Dipongkor and K. Moran, "A Comparative Research of Transformer-Based Neural Text Representation Techniques on Bug Triaging," *arXiv preprint arXiv:2310.06913*, 2023.

[9] A. Ajibode, D. Yunwei, and Y. Hongji, "Software Issues Report for Bug Fixing Process: An Empirical Research of Machine-Learning Libraries," *arXiv preprint arXiv:2312.06005*, 2023.

[10] A. Tagra, H. Zhang, G. K. Rajbahadur, and A. E. Hassan, "Revisiting Reopened Bugs in Open Source Software Systems," *arXiv preprint arXiv:2202.08701*, 2022.

[11] H. Jahanshahi and M. Cevik, "S-DABT: Schedule and Dependency-Aware Bug Triage in Open-Source Bug Tracking Systems," *arXiv preprint arXiv:2204.05972*, 2022.

[12] K. K. Gajula, "Deep Neural Forecasting Models for Climate Change Analysis Using Multivariate Time-Series Data," Journal of Computational Analysis and Applications, vol. 33, no. 8, pp. 6356–6368, 2024.

[13] M. Shahimoon, H. Rasheed, D. R. Rao, K. R. Krishnaiah, K. K. Gajula, et al., "Evolving Predictive Models for Customer Churn in Telecommunications," in Proc. 2nd International Conference on New Frontiers in Communication Technologies, 2025.

[14] K. K. Gajula, "Advanced Stress Level Forecasting Using Quantile-Optimized Regression Forest Models," 2025.