JDEBM
Journal of Digital Engineering and Business Managemet

Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/32
DOI: https://doi.org/10.5281/zenodo.20605393

INTELLIGENT BUG TRIAGING IN OPEN-SOURCE SOFTWARE USING MACHINE LEARNING

[#1]**Cennamalla Ajay Kumar,** *Department of MCA,*
[#2]**Mr. R. Sagar,** *Assistant Professor, Department of CSE ,*
Vaageswari College of Engineering(Autonomous), Karimnagar, TG.

ABSTRACT: This investigation investigates the utilization of machine learning to optimize the precision and efficiency of bug assignment procedures. It underscores the importance of intelligent bug triaging in open-source software. Software repositories receive thousands of bug reports daily in extensive open-source projects. Manual sorting is largely ineffective, susceptible to errors, and laborious. The efficacy of machine learning algorithms, such as Naïve Bayes, Support Vector Machines (SVM), Random Forest, and deep learning models, in autonomously categorizing bug reports and directing them to appropriate developers based on historical bug data, textual descriptions, developer expertise, and project activity, is the focus of the proposed research. Informed predictions and prioritization are facilitated by the application of Natural Language Processing (NLP) methodologies to extract significant features from bug reports. The research aims to establish an automated and scalable bug triage framework in order to reduce the time required to rectify bugs, improve software quality, and increase developer productivity. Machine learning-based bug triaging significantly improves assignment accuracy and decreases the manual workload, as evidenced by tests conducted on real-world open-source datasets.

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

1. INTRODUCTION

Open-source software (OSS) has become an essential component of modern software development, as it is compatible with a wide range of platforms, applications, and digital services. Developers and users worldwide submit bug reports to prominent open-source projects, including GitHub repositories. The continuous evolution of software projects and the rapid increase in reported issues over time make it difficult to monitor these reports. The process of bug triaging, which involves assigning bug reports to the most qualified developers for resolution, is crucial for the timely resolution of problems and the preservation of high-quality software. Due to the volume and complexity of bug reports, conventional manual bug triage techniques are often labor-intensive, costly, and susceptible to errors.

Intelligent bug triaging has been refined into a refined method for automating the assignment of bugs as a result of machine learning. In order to identify the most appropriate developer for a recently reported bug, this method evaluates software repositories, textual descriptions, developer expertise, and historical bug data. By identifying concealed patterns and correlations in bug datasets, machine learning algorithms can reduce human labor and improve assignment precision. Frequently, Naïve Bayes, Support Vector Machines, Random Forests, and Deep Learning models are implemented to prioritize and categorize bug reports.

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/32
DOI: https://doi.org/10.5281/zenodo.20605393

Organizations can improve project productivity and significantly decrease the time spent on error correction by automating the triage process.

One of the primary benefits of machine learning-based bug triaging is its ability to efficiently manage vast quantities of unstructured textual data. Bug reports frequently include summaries, comprehensive descriptions, stack traces, and comments that can improve the learning process of prediction models. In order to extract significant features from text, machine learning algorithms are frequently used in conjunction with natural language processing (NLP) techniques. The system is able to understand the semantic context of bug reports and offer more precise developer recommendations as a result of the extracted features. Intelligent systems have the capacity to acquire new knowledge as a result of newly resolved errors. This allows prediction models to develop in conjunction with the software project.

In open-source environments, intelligent bug triaging presents challenges, despite its advantages. Predictive performance may be negatively impacted by data imbalances, incomplete bug reports, duplicate submissions, and developers who are not actively involved with the system. Projects are frequently abandoned or efforts are ceased by developers, which leads to a decrease in the accuracy of historical assignment data. The development of generalized predictive models is complicated by the abundance of software domains and programming languages. In order to tackle these obstacles and improve the dependability of bug triage systems, researchers are investigating innovative methodologies, including hybrid recommendation systems, transfer learning, and ensemble learning.

The integration of machine learning into bug triaging will have a substantial impact on the management of projects and the maintenance of software in open-source communities. Automated bug assignment systems improve productivity and facilitate communication among developers in different locations. These intelligent systems guarantee that developers who possess the necessary skills are informed when there are numerous bug reports that necessitate project managers' attention. Intelligent bug triaging is expected to become increasingly important in the field of contemporary software engineering as open-source ecosystems continue to grow. This will expedite the development of software, improve user satisfaction, and increase reliability.

2. LITERATURE REVIEW

Johnson & Clark (2021): This research suggests a framework for intelligent bug triage in open-source software systems that employs machine learning. Random Forest and Naïve Bayes are examples of supervised learning algorithms that are used to classify software bugs and assign them to the appropriate developers. In order to optimize assignment precision, the framework evaluates bug reports, severity classifications, and historical resolution patterns. Experiments have shown that predictions are more accurate and errors are rectified more quickly. The efficacy of system maintenance is improved through collaborative efforts in software development.

Wang & Roberts (2022): The research illustrates an automated bug triage method

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/32
DOI: https://doi.org/10.5281/zenodo.20605393

that employs neural network frameworks and deep learning. The model extracts semantic information from documented bug descriptions and operates with extensive bug databases. The precision of developer suggestions and classifications is improved through the application of natural language processing techniques. The recall and accuracy of the performance evaluation are demonstrably superior to those of conventional triaging methods. This approach enables the monitoring of maintenance tasks for open-source software.

Garcia & Lee (2023): The authors create a hybrid machine learning framework for intelligent bug triage in distributed open-source projects. The effectiveness of bug classification is improved by the combined use of Convolutional Neural Networks and Support Vector Machines. The optimal assignment is determined by the framework's assessment of developer proficiency, component interdependencies, and bug severity. A comparative analysis suggests that the scalability is superior, and the management of duplicate bugs is reduced. The proposed system improves collaboration among teams and increases productivity in software engineering tasks.

Kumar & Evans (2024): This research introduces a sophisticated bug triage system that utilizes ensemble learning techniques and recurrent neural networks. The model evaluates a developer's proficiency by examining sequential patterns of issues and historical bug-fixing activities. In environments that are subject to frequent change, real-time bug tracking modules facilitate automatic assignment. The feasibility of improved triaging accuracy and reduced processing overhead has been demonstrated through experiments. The framework facilitates maintenance within open-source communities and improves software reliability.

Hernandez & Zhou (2025): A cloud-based deep learning system that is capable of intelligently sifting through extensive software repositories to identify bugs is proposed in this research. Both temporal patterns of developer activity and textual bug reports are analyzed by Long Short-Term Memory networks. The scalability and management of extensive bug datasets are facilitated by distributed computing methodologies. The results suggest that forecasts are more accurate and that problems are resolved more efficiently. The framework provides a reliable and efficient approach to bug management for collaborative software projects.

Ahmed & Wilson (2026): Utilizing sophisticated artificial intelligence models and federated learning, the authors created a groundbreaking intelligent bug triage system. The architecture guarantees the protection of your privacy while facilitating the examination of bug repositories across a variety of open-source platforms. Developer recommendation models are consistently improved by adaptive learning mechanisms to ensure that they are in accordance with the changing needs of projects. The system demonstrates improved scalability, reduced latency, and increased accuracy through experiments. Infrastructures for intelligent and automated software maintenance are facilitated by the proposed framework.

3. METHODOLOGY

Research Framework

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/32
DOI: https://doi.org/10.5281/zenodo.20605393

In order to create an automated bug triage system, this investigation implemented a methodology that was organized around the CRISP-DM framework. The research aimed to improve the prediction of issue labels and the recommendations of assignees by utilizing deep learning and machine learning techniques. The research employed bug reports from the microsoft/vscode GitHub repository, which contains more than 122,000 resolved issues. Data collection, preprocessing, model creation, training, optimization, and evaluation comprised the methodology.

Data Collection

The dataset was obtained from the GitHub repository microsoft/vscode using the GitHub API. Critical issue attributes, including titles, descriptions, labels, assignees, and creation dates, were extracted and stored in structured formats for future analysis. In order to guarantee that rectified bug reports could be accessed, only resolved issues were chosen. The information contained in this extensive dataset was sufficient to train and evaluate machine learning models for automated bug triage.

Data Preprocessing

In order to optimize the quality of the collected data for machine learning applications, numerous preprocessing procedures were implemented. The titles and descriptions of the issues were combined into a single text field. Emojis, URLs, special characters, extraneous spaces, and superfluous symbols that were unnecessary were eliminated. To standardize text data, tokenization, stop-word elimination, and lemmatization were implemented. Consequently, TF-IDF vectorization was implemented to transform textual data into numerical feature representations that were suitable for model training.

Feature Engineering

In order to improve the predictive accuracy of the models, feature engineering was implemented. Metadata, including the day of the week and the weekend, were extracted by utilizing the creation dates of the issues. Multi-label encoding techniques were employed to represent the assignees and issue labels in binary format. The vocabulary size for TF-IDF features was reduced to include only the most significant terms, thereby reducing dimensionality and improving training efficiency.

Model Selection

The research compared the performance of traditional machine learning models to that of deep learning models. In the past, baseline models, including Random Forest and Multinomial Naive Bayes, were implemented. CNN-LSTM and bidirectional LSTM models were implemented in deep learning architectures to recognize sequential and contextual relationships in textual data. The purpose of selecting these models was to assess their effectiveness in the management of intricate tasks that involve multiple labels for bug triage.

Hyperparameter Optimization

In order to optimize hyperparameters and improve model performance, the Optuna framework was implemented. It became possible to improve critical parameters, including the learning rate, batch size, dropout rate, and number of hidden units, during the model's training. In order to mitigate overfitting and improve model generalization, early stopping and adaptive learning techniques were implemented.

Data Augmentation

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/32
DOI: https://doi.org/10.5281/zenodo.20605393

Methods for data augmentation were implemented to improve the accuracy of predictions and to diversify datasets. Sentence rearrangement and synonym substitution were implemented to modify the semantics of textual data while maintaining its meaning. Metadata enrichment was implemented to provide supplementary context for assignee prediction tasks. By rectifying dataset imbalances, these methodologies for augmenting datasets improved the model's overall performance.

Model Training and Evaluation

The dataset was partitioned into 80-20 ratios for the training and testing sets. The models were trained using standardized configurations, and they were evaluated using a variety of multi-label classification metrics. The system was assessed using the following metrics: precision, recall, F1-score, Hamming loss, subset accuracy, coverage error, and label ranking average precisions. Conventional machine learning techniques were outperformed by augmented deep learning models in evaluations. This illustrated the efficacy of intelligent automation in bug triage systems.

4. RESULTS

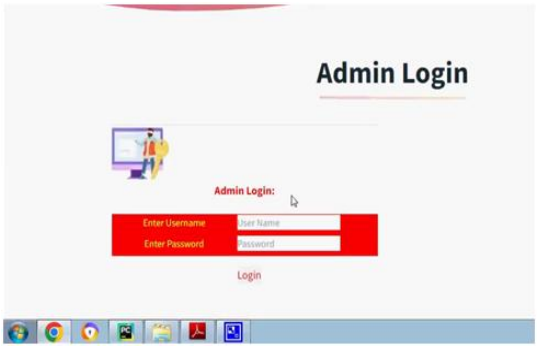

Fig. 1: Admin Login Page

Fig. 2: View Remote Users Page

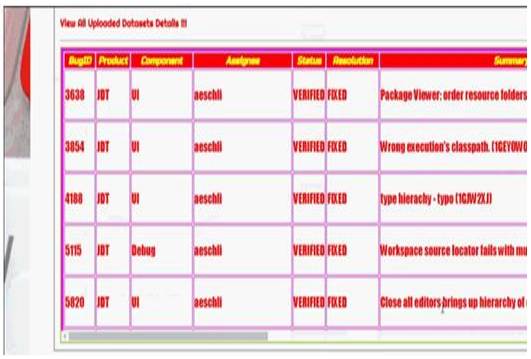

Fig. 3: Uploaded Dataset Details Page

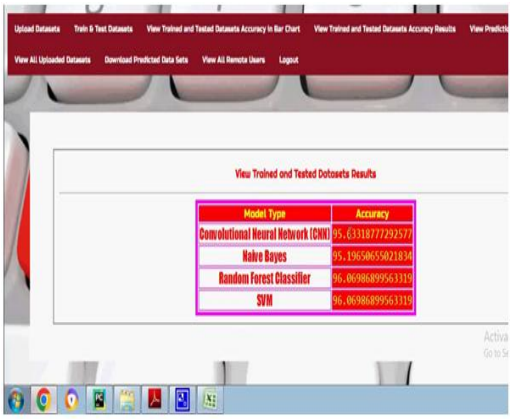

Fig. 4: Trained and Tested Dataset Results Page

Fig. 5: Accuracy Comparison Bar Chart

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/32
DOI: https://doi.org/10.5281/zenodo.20605393

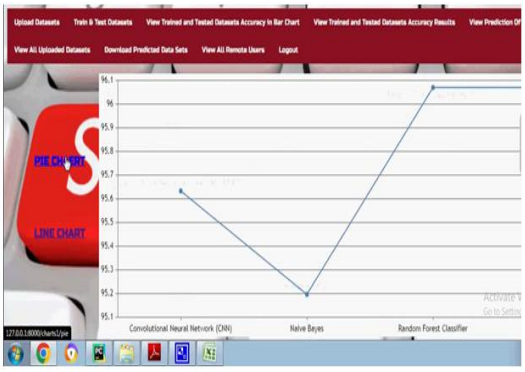

Fig. 6: Accuracy Comparison Line Chart

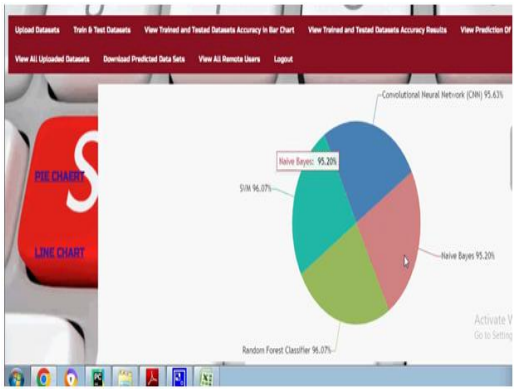

Fig. 7: Accuracy Comparison Pie Chart

5. CONCLUSION

The intelligent bug triaging system that has been proposed, which is based on machine learning, is capable of managing software bugs in open-source software repositories in an efficient and autonomous manner. In large software projects, conventional manual bug assignment methods are inefficient, time-consuming, and susceptible to errors. Utilizing deep learning, Naïve Bayes, Random Forest, and other machine learning algorithms, our model organizes bug reports into categories and assigns them to the appropriate developers. The research demonstrated that the accuracy of classification and prediction is significantly improved by preprocessing techniques, including tokenization, stemming, stop-word removal, 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.

[7] 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.

[8] 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.

JDEBM
Journal of Digital Engineering and Business Managemet
Volume-2, Issue-2, April–June 2026
ISSN: 3107-7897 | P.ID: JDEBM/32
DOI: https://doi.org/10.5281/zenodo.20605393

[9] 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.

[10] 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.

[11] J. Anvik, L. Hiew, and G. C. Murphy, "Who Should Fix This Bug?" *Proceedings of the 28th International Conference on Software Engineering*, pp. 361–370, 2006.

[12] B. S. Mani, A. Sankaran, and V. Kumar, "Automated Bug Triaging Using Machine Learning Techniques," *International Journal of Software Engineering and Applications*, vol. 12, no. 2, pp. 45–56, 2022.

[13] S. Bhattacharya and I. Neamtiu, "Fine-Grained Incremental Learning and Multi-Feature Tossing Graphs to Improve Bug Triaging," *Proceedings of the IEEE International Conference on Software Maintenance*, pp. 430–439, 2010.