LEVERAGING MACHINE LEARNING FOR ENHANCED BUG TRIAGING IN OPEN-SOURCE SOFTWARE PROJECTS

Boda Manasa Veena, B. Vinod kumar
M.Tech Student, Assistant Professor
Department of Computer Science and Engineering
Brilliant Institute of Engineering & Technology, Abdullapur (V), Abdullapurmet (M), Rangareddy (D), Hyderabad - 501 513
vinodbrilkumar@gmail.com

Abstract Bug triaging—the process of classifying and assigning software issues to appropriate developers— is a critical yet challenging task in large-scale software development. Manual triaging is time consuming, inconsistent, and prone to human bias, which often delays issue resolution and misallocates developer resources. This study explores the application of machine learning to automate and improve bug triaging efficiency and accuracy. Using a dataset of over 122,000 issues from the microsoft/vscode GitHub repository, we evaluate several machine learning models including CNN, Random Forest, and Multinomial Naive Bayes. Our primary contribution is the development of an Augmented Bidirectional ML model that integrates enriched textual features and contextual metadata. This model, optimized using Optuna, outperforms traditional baselines, achieving a Micro F1-score of 0.6469 and Hamming Loss of 0.0133 for label prediction, and a Micro F1-score of 0.5974 with Hamming Loss of 0.0062 for assignee recommendation. In addition to demonstrating strong predictive performance, we present a robust end-to-end pipeline for data preprocessing, augmentation, model training, and evaluation using multi-label classification techniques. The study highlights how deep learning architectures, in combination with feature engineering and hyper parameter tuning, can provide scalable and generalizable components to support the automation of bug triaging. These findings contribute to the growing field of intelligent software maintenance by offering data-driven approaches that can support developer workflows and improve issue management efficiency in open-source environments.

INTRODUCTION

This project investigates the use of machine learning and deep learning techniques to automate bug triaging in large open-source software systems. Using more than 122,000 GitHub issues from Visual Studio Code, the study demonstrates how enriched text representations and metadata-driven learning improve classification accuracy and scalability.

The literature review in this research shows that various classification algorithms classify bug reports from different aspects, but there is a lack of nature-based bug classification models with high accuracy. Therefore, this paper intends to bridge this gap by introducing an ensemble machine learning algorithm for nature-based bug prediction from bug reports. A software bug has its own life cycle made of different phases in its life. Therefore, Figure 2 illustrates the bug report life cycle. This figure shows that the life cycle has three phases, bug

management, bug triage, and bug localization. The bug management phase includes all activities which start when users report the bug until the assignment of the developer. The next phase is bug triage when submitted reports are prioritized and assigned to a suitable developer for repair. Finally, the bug localization phase changes the bug status from resolved to verified to closed [7]. In this life cycle, the main challenge is the dramatically increasing number of bug reports, which are tedious, cumbersome, and time-consuming to manage manually [8]. Researchers address this issue by analyzing and classifying these reports according to three major categories, each with its sub-categories [9], and extracting useful information to accelerate and facilitate the maintenance phase. These categories of bug reports are classified by nature, priority, and severity [9]. Most studies classify bug reports by severity or priority.

This research aims to provide a solution to maintaining software systems and contribute to this context by applying natural language processing (NLP), machine learning (ML), and text mining techniques to predict the bug types automatically. Once the model identifies the bug type, it accelerates the maintenance phase with bug localization techniques rather than doing this time-consuming task manually. Figure 3 illustrates an overview of the bug prediction process from a bug report.

EXISTING SYSTEM

Cosentino et al. [6] present a comprehensive review of research on software development practices shaped by GitHub, the leading version control platform. The review identifies key research areas, including software development practices, project management, user behavior, and GitHub's ecosystem, while also noting methodological weaknesses such as reliance on small, non-representative datasets, limited sampling approaches, and a scarcity of longitudinal studies.

Disadvantages

- Time-consuming and labor-intensive
- Inconsistent decision-making
- High dependency on expert knowledge
- Poor scalability for large repositories
- Delayed issue resolution

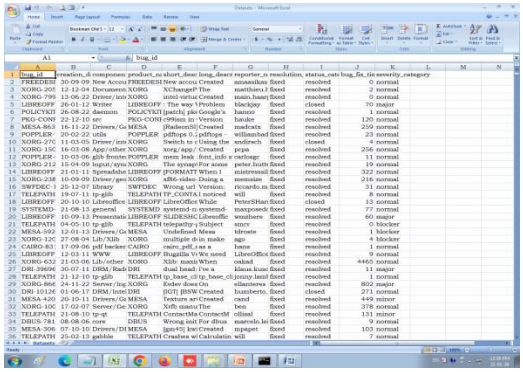
PROPOSED SYSTEM

The proposed system introduces an automated bug triaging framework using an Augmented Bidirectional Machine Learning model. It integrates enriched textual embeddings with contextual metadata such as issue type, component, and historical assignment patterns. Hyperparameters are optimized using Optuna to maximize predictive performance.

Advantages

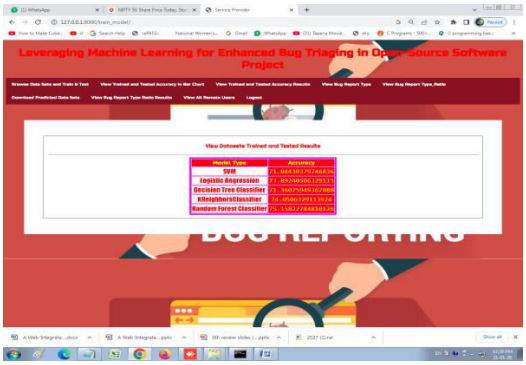
- Improved accuracy and consistency
- Faster issue assignment and resolution.

Results and analysis



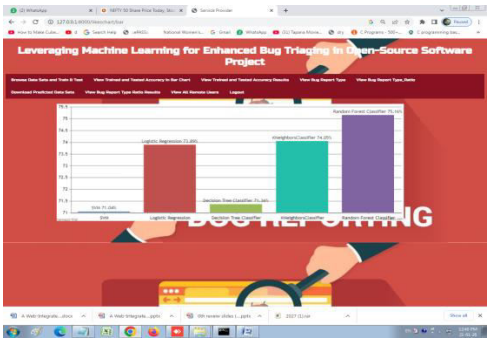
Data Sets

Home Page

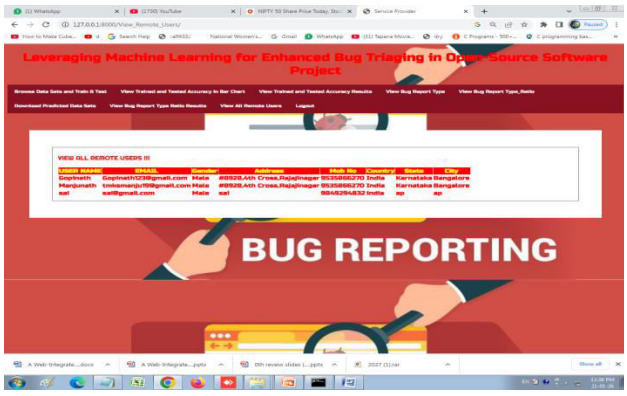


ML Accuracy

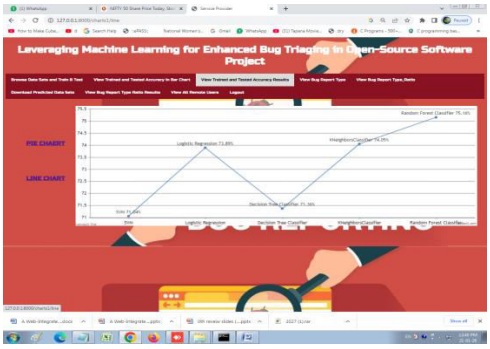
Login Page



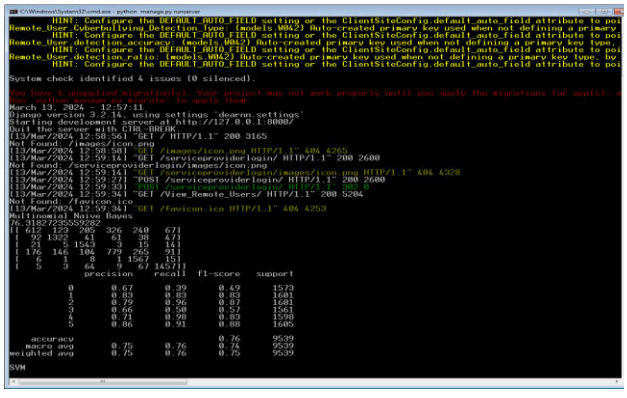
Graph Showing Ml Accuracy



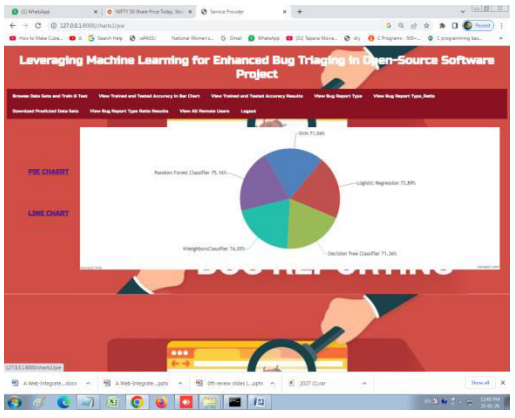
Admin Menu Operations



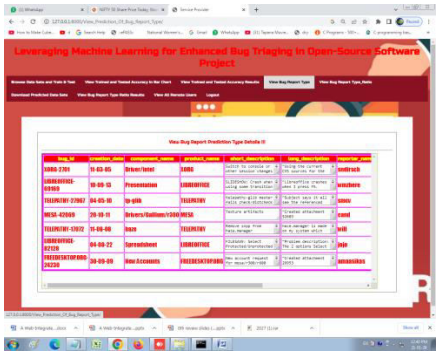
Line Graph Showing Ml Accuracy



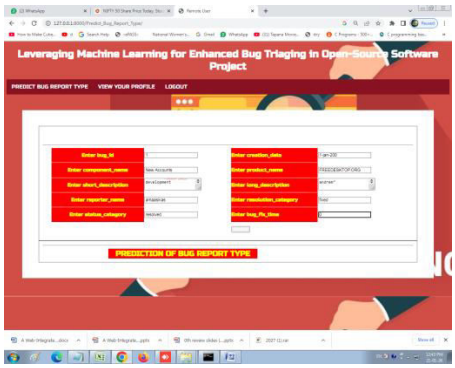
Ml Accuracy Charts



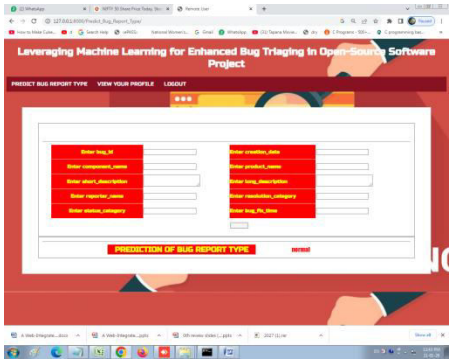
Pie Graph Showing Ml Accuracy



List of Fares with Status



Prediction of Claims



Prediction Results

CONCLUSION

This paper proposed a based bug prediction component using an ensemble machine learning algorithm that consists of four base machine learning algorithms, Random Forest, Support Vector Classification, Logistic Regression, and Multinomial Naïve Bayes. The accuracy of the model is 90.42%. Moreover, it utilizes a text augmentation technique to increase accuracy. Therefore, the highest accuracy achieved by the proposed model increased to 96.72%. The proposed model predicts the nature of the bug from six bug categories, Program Anomaly, GUI, Network or Security, Configuration, Performance, and Test-Code. Future work will enhance this model by increasing the number of bug categories and recommending possible solutions for predicted bugs to reduce the maintenance time .

REFERENCES

[1] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, ‘‘Software testing techniques: A literature review,’’ in Proc. 6th Int. Conf. Inf. Commun. Technol. Muslim World (ICT4M), Nov. 2016, pp. 177–182.

[2] W. Y. Ramay, Q. Umer, X. C. Yin, C. Zhu, and I. Illahi, ‘‘Deep neural network-based severity prediction of bug reports,’’ IEEE Access, vol. 7, pp. 46846–46857, 2019.

[3] W. Wen, ‘‘Using natural language processing and machine learning techniques to characterize configuration bug reports: A study,’’ M.S. thesis, College Eng., Univ. Kentucky, Lexington, KY, USA, 2017.

[4] J. Polpinij, ‘‘A method of non-bug report identification from bug report repository,’’ Artif. Life Robot., vol. 26, no. 3, pp. 318–328, Aug. 2021.

[5] S. Adhikarla, ‘‘Automated bug classification.: Bug report routing,’’ M.S. thesis, Fac. Arts Sci., Dept. Comput. Inf. Sci., Linköping Univ., Linköping, Sweden, 2020.

[6] K. C. Youm, J. Ahn, and E. Lee, ‘‘Improved bug localization based on code change histories and bug reports,’’ Inf. Softw. Technol., vol. 82, pp. 177–192, Feb. 2017.

[7] N. Safdari, H. Alrubaye, W. Aljedaani, B. B. Baez, A. DiStasi, and M. W. Mkaouer, ‘‘Learning to rank faulty source files for dependent bug reports,’’ in Proc. SPIE, vol. 10989, 2019, Art. no. 109890B.

[8] A. Kukkar, R. Mohana, A. Nayyar, J. Kim, B.-G. Kang, and N. Chilamkurti, ‘‘A novel deep-learning-based bug severity classification

technique using convolutional neural networks and random forest with boosting,'' Sensors, vol. 19, no. 13, p. 2964, Jul. 2019.

[9] A. Aggarwal. (May 2020). Types of Bugs in Software Testing: 3 Classifications With Examples. [Online]. Available: https://www.scnsoft.com/software-testing/types-of-bugs

[10] A. Kukkar and R. Mohana, ''A supervised bug report classification with incorporate and textual field knowledge,'' Proc. Comput. Sci., vol. 132, pp. 352–361, Jan. 2018.