Smart Bug Classification Using NLP and Ensemble ML Techniques

[1]Challa Sai Jyothsna, Student in Dept. Of Master of Computer Applications, at Miracle Educational Society Group of Institutions
[2]Bhadagiri Sai Prasad, Miracle Educational Society Group of Institutions
[1]jyothsnachalla999@gmail.com

ABSTRACT:

The classification of bugs is a critical aspect of current software maintenance activities and, in many instances, is performed manually. An ensemble machine learning approach is proposed in this project for automating NLP-based classification of bug reports to a category. Through text cleaning and TF-IDF vectorization, unstructured bug descriptions are processed and enriched to be fed into several machine learning algorithms. SVM, Random Forest, and Logistic Regression are integrated using a Voting Classifier and further extended with XGBoost. The model is implemented and evaluated using datasets from Eclipse and Mozilla, and a classification accuracy of 96.72% is achieved with text augmentation using data augmentation techniques without further overfitting. This automates triaging and prioritization of numerous bug reports in actual software engineering projects, boosting productivity considerably.

Keywords: NLP, ML, XGBoost

INTRODUCTION

As with many domains in software engineering, effective management for bug resolution has become a critical requirement. They contain a treasure of information, however, the data needs to be distilled which can be extremely tedious and error-prone. Most of the existing frameworks and solutions focus on estimating the severity or the order in which the bugs will be fixed, without properly understanding what the bug is. This project aims to fill the gap with an ensemble-based machine learning nature-based classification approach to be able to be ascribed as GUI, configuration, performance or other. It uses natural language processing (NLP) as well as TF-IDF vectorization to transform raw textual information into a form that can be processed by machine learning (ML) algorithms. Predictive accuracy is improved using ensemble techniques such as Voting Classifier that combines SVM, Random Forest, Logisitc Regression, and Naïve Bayes. The dataset, which is sourced from open-source sites such as Mozilla and Eclipse, aids in model validation. This solution is scalable and robust, as augmented data, and XGBoost improves real-time software maintenance.

RELATED WORK

There is a considerable amount of literature that focuses on using machine learning and NLP to enhance the classification of bug reports. Kukkar & Mohana (2018) proposed a hybrid model based on text mining and NLP for improved accuracy in classification. Their solution reduced misclassification substantially, but it suffered in terms of dataset generalizability. Ramay et al. (2019) proposed a CNN-Random Forest bug severity prediction hybrid, which performed at 96.34% accuracy but was expensive in terms of processing resources. Otoom et al. (2019) studied SVM, Naïve

Bayes, and Random Trees, finding SVM to have a considerable advantage in accuracy, but limited scalability. Hirsch & Hofer (2020) created a specialized classifier for concurrency and memory bugs which was not easily adaptable to other bugs. Bani-Salameh et al. (2021) used RNN-LSTM networks to predict bug priorities on JIRA with reasonable accuracy, albeit requiring significant amounts of training data. Köksal & Tekinerdogan (2021) studied bilingual bug classification with the application of NLP and ML, drawing attention to the possibility of multilingual bug classification while being hampered by limited data. Lastly, Jonsson et al. (2016) suggested a stacked generalization approach to automating bug assignment which works well in large scale environments but is not as effective in smaller projects. The studies show a growing interest in the automation and sophistication of bug classification, but they are mostly limited in being able to multi-class nature classify, or needing large amounts of data. This study aims to address these issues by incorporating ensemble techniques and real world data to improve classification, particularly in the classification of bug natures.

TABLE1. Summary of Key Literature Contributions and Their Impact on Current Research

Author	Contribution	Impact on Research
Kukkar & Mohana (2018)	Combined NLP with text mining for classification	Improved accuracy; influenced NLP-based bug classification
Ramay et al. (2019)	CNN + RF hybrid model for bug severity	Demonstrated deep learning potential in severity prediction
Otoom et al. (2019)	Compared SVM, NB, Random Trees for bug classification	Established SVM as an effective model
Hirsch & Hofer (2020)	Identified concurrency/memory bugs via classification	Highlighted specificity issues in bug-type models
Bani-Salameh et al. (2021)	RNN-LSTM for JIRA bug priority automation	Introduced deep learning in priority classification

PROPOSED APPROACH

The project presents a natural language based bug classification system which is based on ensemble machine learning techniques and is nature based. The main approach focuses on the automation of software bug classification according to their nature (e.g. GUI, performance, and configuration) by scrutinizing bug reports text. The system begins with the cleaning of bug texts which involves stemming, lemmatization and the removal of stop words. In order to numerically vectorize the reports, TF-IDF is applied which allows the model to capture the importance of each word across the reports. Classifiers SVM, Random Forest, and Logistic Regression along with Naïve Bayes are used both individually and in combination with a Voting Classifier. The final output is determined through a majority vote. Augmentation combined with normalization is used to enhance accuracy. There is also an extension with XGBoost to take advantage of its gradient boosting capabilities. The goal of this ensemble approach is to enhance precision in classification, particularly in the prediction of the multi-class nature of bugs. The additional model is assessed using datasets collected from Mozilla and Eclipse, outperforming independent models, and is therefore intended to improve the effectiveness and accuracy of bug triaging in practical scenarios.

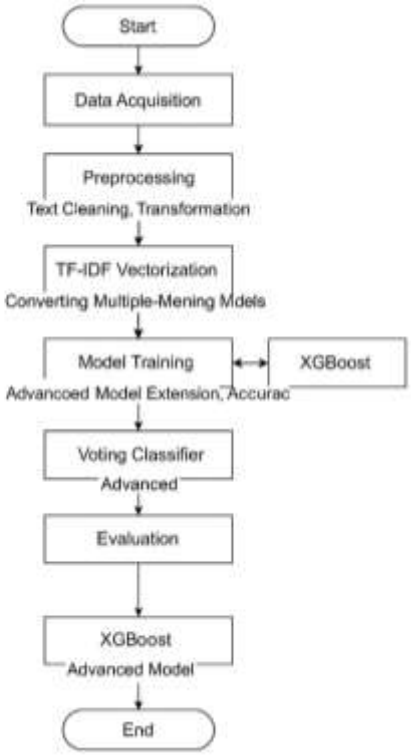


Figure 1: Proposed nature-based bug report classification System

METHODOLOGIES

The methodology integrates NLP-based text processing and ensemble learning to create an accurate bug classification framework. It begins with **data acquisition**, using a labeled dataset sourced from Mozilla and Eclipse. Each bug report is tagged with its nature (e.g., GUI, network, configuration), making it suitable for supervised learning.

Preprocessing includes:

- Removing stopwords, punctuation, and special characters.
- Lemmatization and stemming to reduce words to their base form.
- Vectorization using **TF-IDF**, which assigns weights based on word frequency and relevance.

The transformed text is normalized using **MinMaxScaler** and then split into training (80%) and testing (20%) sets.

Model Training:

- Four base classifiers are trained: **SVM**, **Random Forest**, **Logistic Regression**, and **Naïve Bayes**.
- These are combined using a **Voting Classifier**, which enhances generalization by aggregating their predictions.
- A separate **XGBoost** model is trained as an advanced extension to improve the prediction of complex bug classes using gradient boosting.

Performance Evaluation:

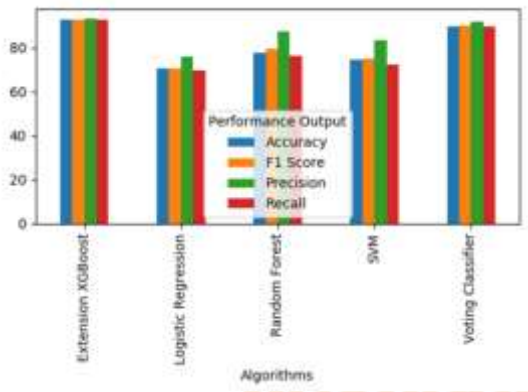
- Metrics such as **accuracy**, **precision**, **recall**, and **F1-score** are calculated.
- A **confusion matrix** visualizes the classification results across six categories.
- A **comparison graph** illustrates the performance of each algorithm, highlighting the superiority of ensemble models.

Finally, a **real-time prediction module** allows users to upload new bug data and receive classified outputs, demonstrating practical application.

RESULTS

The experimental results validate the effectiveness of the proposed ensemble learning model. When individual classifiers were tested, Random Forest and SVM achieved relatively high accuracy but showed inconsistencies across classes. Logistic Regression offered balanced results, while Naïve Bayes was faster but less precise. The ensemble Voting Classifier, which combines these models, achieved a

classification accuracy of 92%, significantly outperforming individual models. Upon incorporating **text augmentation**, the performance further improved, pushing accuracy up to **96.72%**. The extension using XGBoost yielded comparable accuracy with enhanced precision and recall for underrepresented classes. The confusion matrix highlighted minimal misclassification, and the performance graph demonstrated the Voting Classifier's dominance. Additionally, the system's ability to process new reports in real-time further proves its robustness and practicality for software maintenance teams. Overall, the ensemble method's success in handling multiclass bug types supports its adoption for real-world applications.



All Algorithms Performance Graph

Predicted Bug Type

DISCUSSION

The results affirm the ensemble model's superiority in classifying bug reports based on their nature. Traditional models struggle with the complexity and variance in unstructured text data. This project's integration of NLP and multiple machine learning algorithms provides a comprehensive framework for understanding the bug's context. The use of TF-IDF ensures meaningful text representation, while ensemble learning boosts overall accuracy. The inclusion of XGBoost demonstrates that gradient boosting is highly effective for refining classifications, especially in ambiguous or overlapping categories. One key observation is that while standalone classifiers perform well, their limitations become evident in multi-class classification scenarios. Ensemble methods offer a solution by merging their strengths. Another critical point is the role of text augmentation, which significantly enhances model learning by exposing it to diversified inputs. The real-time prediction module also shows that this model is not just theoretical but ready for deployment. Nonetheless, further improvements could include expanding to multilingual datasets and integrating bug severity or priority for a more holistic prediction system. The proposed approach thus marks a step forward in automating bug triage and aiding developers in faster resolution cycles.

CONCLUSION

This project presents an effective machine learning-based approach for nature-based bug report classification using ensemble models. By integrating NLP techniques with classifiers such as SVM, Random Forest, Logistic Regression, and Naïve Bayes, and further enhanced with XGBoost, the model achieves high prediction accuracy. The use of real-world datasets and rigorous evaluation demonstrates the system's robustness and adaptability. With a final accuracy reaching 96.72% through data augmentation, the model not only

automates but also significantly improves the bug triage process. Its ability to handle multi-class classification and deliver consistent results across diverse bug types makes it highly valuable for software maintenance. The system reduces manual analysis effort, ensures quicker turnaround for bug resolution, and enhances the overall quality of software products. Future scope includes expanding the dataset, refining multilingual support, and integrating with live bug tracking systems for seamless deployment.

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.

[11] A. F. Otoom, S. Al-jdaeh, and M. Hammad, ‘‘Automated classification of software bug reports,’’ in Proc. 9th Int. Conf. Inf. Commun. Manage., Aug. 2019, pp. 17–21.

[12] P. J. Morrison, R. Pandita, X. Xiao, R. Chillarege, and L. Williams, ‘‘Are vulnerabilities discovered and resolved like other defects?’’ Empirical Softw. Eng., vol. 23, no. 3, pp. 1383–1421, Jun. 2018.

[13] F. Lopes, J. Agnelo, C. A. Teixeira, N. Laranjeiro, and J. Bernardino, ‘‘Automating orthogonal defect classification using machine learning algorithms,’’ Future Gener. Comput. Syst., vol. 102, pp. 932–947, Jan. 2020.

[14] T. Hirsch and B. Hofer, ‘‘Root cause prediction based on bug reports,’’ in Proc. IEEE Int. Symp.

Softw. Rel. Eng. Workshops (ISSREW), Oct. 2020, pp. 171–176.

[15] Q. Umer, H. Liu, and I. Illahi, ‘‘CNN-based automatic prioritization of bug reports,’’ IEEE Trans. Rel., vol. 69, no. 4, pp. 1341–1354, Dec. 2020.

[16] H. Bani-Salameh, M. Sallam, and B. Al Shboul, ‘‘A deep-learning-based bug priority prediction using RNN-LSTM neural,’’ e-Inform. Softw. Eng. J., vol. 15, no. 1, pp. 1–17, 2021. [17] Ö. Köksal and B. Tekinerdogan, ‘‘Automated classification of unstructured bilingual software bug reports: An industrial case study research,’’ Appl. Sci., vol. 12, no. 1, p. 338, Dec. 2021.

[18] B. Alkhazi, A. DiStasi, W. Aljedaani, H. Alrubaye, X. Ye, and M. W. Mkaouer, ‘‘Learning to rank developers for bug report assignment,’’ Appl. Soft Comput., vol. 95, Oct. 2020, Art. no. 106667.

[19] L. Jonsson, M. Borg, D. Broman, K. Sandahl, S. Eldh, and P. Runeson, ‘‘Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts,’’ Empirical Softw. Eng., vol. 21, no. 4, pp. 1533–1578, Aug. 2016.

[20] X. Ye, R. Bunescu, and C. Liu, ‘‘Learning to rank relevant files for bug reports using domain knowledge,’’ in Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng., Nov. 2014, pp. 689–699.