International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 07 | July - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Advanced Bug Report Prediction Through Nature-Based Ensemble Machine Learning

Abhijeet. Premanand. Naik[1], Dr. T Vijaya Kumar[2]

[1] *Student, Department of MCA, Bangalore Institute of Technology, Karnataka, India*
[2]*Professor & Head, Department of MCA, Bangalore Institute of Technology, Karnataka, India*

Abstract - In today's software development world, quickly and accurately classifying bug reports is crucial for fixing issues on time and keeping software quality high. Bugs are a vital part of this process, carrying detailed information such as description, status, reporter, assignee, priority, severity, among others. It becomes tiresome and time-consuming to analyze these reports manually in search of all defects, especially when the size of these reports increases. Thus, automation is one of the best solutions. While current research is focused mainly on automating the identification of bug severity or priority, they often forget identification based on the nature of the bugs, which requires multi-class classification. A new prediction model has been used that analyzes BRs for the prediction of nature of the bugs. This model will combine the ensemble ML algorithm with NLP and ML techniques. In particular, the model is tested against publicly available datasets from Mozilla and Eclipse, which further categorize the bugs into six types: program anomaly, GUI, network or security, configuration, performance, and test-code. The results indicate that our model gave significantly higher accuracy compared to many pre-existing models without text augmentation, at 90.42 percent and with text augmentation at 96.72 percent.

Keywords: Natural language Processing (NLP), Machine Learning (ML), Bug Reports (BRs)

I. INTRODUCTION

In the modern digital age, software systems are important to many aspects of our life and advance technology throughout a broad spectrum of industries. Software becomes increasingly difficult to maintain dependable and functional as it becomes more complicated. Finding defects in software is a significant problem that may have an impact on both the program's functionality and user experience. As a solution to this issue, our project sorts and detects software faults using state-of--art machine learning techniques with the goal of improving software maintenance. Software bugs are difficult to prevent and this might result in anything from minor glitches to catastrophic malfunctions. These problems may jeopardize the security and functionality of the software. Although conventional bug prediction techniques are helpful, they frequently cannot keep up with the intricacy of today's software. To increase the accuracy and dependability of bug prediction, our project combines ML techniques with NLP. These integrated methods leverage each model's strengths by combining multiple learning algorithms. More accurate projections are produced by this collaborative method than by using a single model alone.

Our methodology focuses on classifying six report categories Program Anomaly GUI, Network or Security, Configuration, Performance, and Test-Code based on type of bug. By addressing a broad spectrum of software bugs, this grouping improves prediction accuracy. Our model integrates both NLP and ML techniques to build a unified algorithm that evaluates reports of bug and identifies the various vulnerabilities. We use the public datasets from two popular online bug databases, Eclipse and Mozilla, to test our proposed methodology. These datasets provide a multitude of reports that are listed into the six previously described classes. Our methodology entails compiling and sanitizing bug report data, followed by the identification and selection of the critical components most influential in the incidence of bugs.

While simultaneously speeding up the resolution process, the suggested method automatically categorizes the different kinds of issues based on text and context. In addition to saving manual work, this automated classification increases triage and resolution speed and accuracy. This kind of model concentrates based on the attributes of the bug; this aids in setting priorities and, consequently, in assigning the bug to the developer, which results in the advantages of addressing key concerns first. Furthermore, the automation and simplification of the problem management procedure itself ought to result from the integration of the suggested model with the current bug tracking tools. As soon as a new bug report is submitted, it can help with the real-time classification process, providing instant insight into the type of bug. Development teams' workflow will be substantially streamlined as a result, allowing them to focus more on resolution than on completing classification processes.

II. LITERATURE SURVEY

Goyal et al. investigated the five ensemble-based categorization strategies of bug reports and showed that, compared to traditional ML approaches, these collective methods greatly improve the procedure for matching each report of bug to most suitable developer [1]. Patil et al. used NLP to automatically label bugs reports by identifying duplicates in bug report and examining both their structured and unstructured features. This method finds duplicates before they are added to the system, which lowers operating expenses and eliminates the creation of redundant data [2]. Aburakhia et al. provided a method Based on ML to Establish the Default Level of Bug Severity, according to recent studies the default severity rating assigned to the majority of bug reports in these systems sometimes misrepresents the true

severity of bug. Aburakhia et al. suggested a new technique to automatically label default bug reports as non-severe or severe to deal with this issue [3]. Shatnawi et al. work presents a fully automated Machine Learning model to estimate problem severity and priority in the Eclipse dataset. [4]. Several metrics, such as component name, summary, assignee, and reporter, are derived from bug reports in closed-source projects using the JIRA bug-tracking system and are included in the model. High accuracy bug priority level prediction is made possible by use of the 5-layer deep learning RNN-LSTM neural network [5]. Capbug: An Automated System for classifying and prioritizing Bug Reports provide a brand-new framework called CaPBug that is intended to automatically classify and rank bug reports. In order to put the concept into practice, Ahmed et al. examined more than 2,000 bug reports from the bug repositories to create baseline corpus with six categories and five priority levels [6]. Immaculate et al. in his study used Random Forest ensemble classifiers to be able to increase prediction accuracy and reliability in various software development environment while validating the K-Fold cross-validation. This approach's byproducts will improve software quality and save development resources by minimizing the result of errors on system coherence and performance during later phases of development [7]. Yadav et al. used HAN model is able to utilize attention methods to capture hierarchical relationships inside Bug reports. Rich functionality in text normalization from the DistilBERT tokenizer and fine-grained comprehension from bidirectional GRU layers make it possible [8]. Zheng et al. used text that is represented as a Text CNN and contains a bug report. The average experimental findings across five different datasets show that CorNER achieves significant improvement, attesting to 6.24% F1-Score [9]. A recommendation engine is being developed in a different study to effectively allocate developers who possess the necessary expertise to handle problem complaints in software repositories [10]. Lamkanfi et al. study contains reports of bugs from Eclipse Platform, JDT, CDT, PDE, Core, Firefox, Thunderbird, Bugzilla [11]. Kukkar et al provided a hybrid approach with the goal of correctly classifying reports of bugs as either bugs or non-bugs. Paper uses the BRs to incorporate four more textual fields and makes use of Bigram feature extraction techniques and TF-IDF [12]. Hirsch et al. in the study used dataset spanning over 70 projects, together with unique preprocessing methods, to assess and train the models. Models with macro average F1 scores as high as 0.69% indicate that they have potential to be used as efficient bug classifiers in a various project [13]. Comprehensive analysis of DBRP models for each of the five main BRP tasks. Papers from global databases released between 2015 and 2023 are included in the review. They are graded according to how well they use deep neural networks, word embedding and text representation models, and bug report feature extraction approaches [14]. Using heuristics, Thomas et al. gathered and analyzed 54,755 closed bug reports from the issue trackers of 103 GitHub projects to produce a benchmark dataset of 10,459 complaints. Based on the main problem of semantic, memory, and concurrency, a subset of this dataset was manually categorized into three groups. A NLP algorithm used these reports as input, evaluating many classifiers for root cause prediction [15].

III. EXISTING SYSTEM

In many software development environments, bug report classification is often a manual process. Developers and testers read through bug reports, categorize them based on their understanding, and then assign them to the appropriate team for resolution. This process is not only time-consuming but also prone to human error and inconsistency. Automated systems for bug report classification do exist, but they typically rely on single machine learning algorithms. These systems use text classification techniques to analyze the content of bug reports and predict the category of the bug. However, these single-classifier systems often face limitations in accuracy and efficiency. They may struggle with the diverse and complex nature of bug reports, leading to incorrect classifications and delayed issue resolution.
Moreover, existing systems generally do not incorporate advanced techniques like text augmentation to enhance prediction accuracy. As a result, the effectiveness of these systems is limited, and they may not perform well across different datasets or adapt to new types of bugs reports easily.

Disadvantages

- A hybrid deep learning detection policy to increase the efficacy and efficiency of report generation is absent from the current systems.
- Their attempt to classify problem reports based on nature categories is unsuccessful. which produce better accurate forecasts.

IV. PROPOSED SYSTEM

This approach integrates ML, NLP and text mining approaches to achieve optimal accuracy and efficacy in bug prediction. In order to find the best classifiers for automatic nature-based bug report classification, the system first evaluates various approaches. Through the identification and application of the top-performing classifiers, the system establishes a solid basis for precise bug prediction. Using an ensemble machine learning approach, the suggested system builds on this basis. By combining the benefits of many classifiers, this method enhances overall prediction performance and guarantees a solid and trustworthy classification procedure.

Advantages:

- By using multiple machines learning base classifiers trained on a benchmark dataset, the proposed technique improves nature-based bug prediction and ensures reliable and accurate results.
- A thorough and efficient bug prediction model is produced by combining ML, NLP, and text-mining techniques.

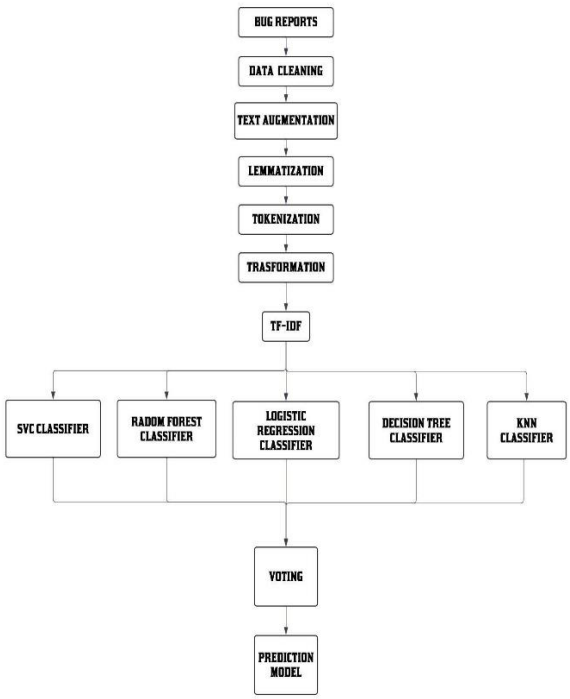

Fig -1: Proposed Model

V. IMPLEMENTATION

Bug Reports Collection:
We use public datasets from sources like Mozilla and Eclipse to train our model. These datasets provide the information needed to teach our system how to classify bug reports accurately. Each report includes at least a title, a description, status, and metadata.

Data Cleaning:
Data cleaning is a process to turn raw bug reports into a format suitable for processing. These techniques involve Removing Noise, Text Normalization, Stop Words Removal.

Text Augmentation:
Text augmentation enriches the dataset and strengthens models by artificially creating more data. It generates new variations of existing reports to expand the dataset.

Lemmatization:
Lemmatization lowers all words to their basic form or root, which contributes to the size reduction of the text.

Tokenization:
Tokenization transforms the cleaned and lemmatized text into separate tokens or words. Here, this process involves the application of tokenization tools for breaking the sentences down into words or subwords.

Transformation:
Next, these tokens are changed into numerical forms so that ML processing can be done.

TF-IDF:
This comes with two metrics: Term Frequency and, Inverse Document Frequency. The first one, Term Frequency (TF), looks at the occurrences of the word within a document. The second, Inverse Document Frequency (IDF), measures how rarely a word occurs across all documents.

Voting Classifier
To develop more accurate prediction model, the Voting Classifier shall be used. The work is that the predictions by individual classifiers are aggregated inside the voting classier such that the final prediction benefits from the strengths of all the classifiers participating in this process.

Prediction Model
Again, these results obtained by the voting classifier in the report are predefined categories such as Program Anomaly, GUI, Network or Security, Configuration, Performance, or even Test-Code. In this context, accuracy will be appraised based on the metrics defined as follows: accuracy, precision, recall, F1-score, and confusion matrix. This will ensure that the model generalizes well on unseen data using cross-validation**.**

VI. CONCLUSION

In this study, we have combined ML learning with NLP to create a potent system that can predict type of bug. Six major categories are used to categorize the issue reports: Program Anomaly, GUI, Network or Security, Configuration, Performance, and Test-Code increase bug precision. We discovered that using datasets from Mozilla and Eclipse gave us a strong basis on which to train and evaluate our models. Several ml classifiers were integrated into ensemble learning, which has the potential to improve prediction accuracy. By addressing the shortcomings we saw in previous research, our novel text augmentation methodology has also helped us increase the strength of our system. Results of test fully demonstrated the effectiveness of our strategy, which could prove to be useful tool for project managers and software engineers alike. Our model is designed to overcome the limitations of existing systems, which often rely on a single classifier and struggle with accuracy. By using an ensemble approach, our system combines the strengths of different classifiers, leading to better performance. Testing with a benchmark dataset shows that our model is more accurate and efficient at predicting bugs compared to traditional methods

VII. FUTURE ENHANCEMENTS

Subsequent improvements to the present problem classification system can be made to make it more adaptive and proactive. By using models like transformers and RNNs, bug reports' textual data can be better understood, perhaps leading to an increase in classification accuracy. Adding more open-source projects to the dataset will increase the model's sturdiness and ability to handle a variety of inputs. By adding more categories, defects can be identified and categorized in more depth, increasing the coverage and specificity of the system. Furthermore, suggesting potential workarounds for

anticipated faults can help developers by offering potential patches associated with categories. Lastly, the process will be streamlined by linking the bug prediction system with development platforms like GitLab, GitHub, or JIRA, which will improve the tool's usability and convenience for developers.

VIII. REFERENCES

[1] Goyal, Anjali, and Neetu Sardana. "Empirical analysis of ensemble machine learning techniques for bug triaging." 2019 Twelfth International Conference on Contemporary Computing (IC3). IEEE, 2019.

[2] Patil, Avinash, and Aryan Jadon. "Auto-labelling of bug report using natural language processing." 2023 IEEE 8th International Conference for Convergence in Technology (I2CT). IEEE, 2023.

[3] Aburakhia, Abdalrahman, and Mohammad Alshayeb. "A Machine Learning Approach for Classifying the Default Bug Severity Level." Arabian Journal for Science and Engineering (2024): 1-18.

[4] Shatnawi, Mohammed Q., and Batool Alazzam. "An Assessment of Eclipse Bugs' Priority and Severity Prediction Using Machine Learning." International Journal of Communication Networks and Information Security 14.1 (2022): 62-69.

[5] Bani-Salameh, Hani, Mohammed Sallam, and Bashar Al shboul. "A deep-learning-based bug priority prediction using RNN-LSTM neural networks." e-Informatica Software Engineering Journal 15.1 (2021).

[6] Ahmed, Hafiza Anisa, Narmeen Zakaria Bawany, and Jawwad Ahmed Shamsi. "Capbug-a framework for automatic bug categorization and prioritization using nlp and machine learning algorithms." IEEE Access 9 (2021): 50496-50512.

[7] Immaculate, S. Delphine, M. Farida Begam, and M. Floramary. "Software bug prediction using supervised machine learning algorithms." 2019 International conference on data science and communication (IconDSC). IEEE.

[8] Yadav, Anurag, and Santosh Singh Rathore. "A Hierarchical Attention Networks based Model for Bug Report Prioritization." Proceedings of the 17th Innovations in Software Engineering Conference. 2024.

[9] Zheng, Wei, et al. "Duplicate Bug Report detection using Named Entity Recognition." Knowledge-Based Systems 284 (2024): 111258.

[10] Al-Bayati, Jalal Sadoon Hameed, Mohammed Al-Shamma, and Furat Nidhal Tawfeeq. "Enhancement of Recommendation Engine Technique for Bug System Fixes." Journal of Advances in Information Technology 15.4 (2024).

[11] Lamkanfi, Ahmed, Javier Pérez, and Serge Demeyer. "The eclipse and mozilla defect tracking dataset: a genuine dataset for mining bug information." 2013 10th Working Conference on Mining Software Repositories (MSR). IEEE, 2013.

[12] Kukkar, Ashima, and Rajni Mohana. "A supervised bug report classification with incorporate and textual field knowledge." Procedia computer science 132 (2018): 352-361.

[13] Hirsch, Thomas, and Birgit Hofer. "Using textual bug reports to predict the fault category of software bugs." Array 15 (2022): 100189.

[14] Ahmad, Aminu Abdullahi, et al. "Deep Bug Reports Processing (DBRP): A Systematic Literature Review." (2023).

[15] Thomas, and Birgit Hofer. "Root cause prediction based on bug reports" IEEE, 2020.