E-ISSN: 2707-6644
P-ISSN: 2707-6636
www.computersciencejournals.com/ijcpdm
IJCPDM 2024; 5(2): 18-22
Received: 15-06-2024
Accepted: 22-07-2024

K Kumaraswamy
Assistant Professor, Department of Information and Technology, Malla Reddy Engineering College for Women, Autonomous, Hyderabad, Telangana, India

Barla Sreya
Student, Department of Information and Technology, Malla Reddy Engineering College for Women, Autonomous, Hyderabad, Telangana, India

B Keerthi
Student, Department of Information and Technology, Malla Reddy Engineering College for Women, Autonomous, Hyderabad, Telangana, India

B Sindhuja
Student, Department of Information and Technology, Malla Reddy Engineering College for Women, Autonomous, Hyderabad, Telangana, India

Corresponding Author:
K Kumaraswamy
Assistant Professor, Department of Information and Technology, Malla Reddy Engineering College for Women, Autonomous, Hyderabad, Telangana, India

An xgboost-based model for predicting nature-based bug reports

K Kumaraswamy, Barla Sreya, B Keerthi and B Sindhuja

DOI: https://doi.org/10.33545/27076636.2024.v5.i2a.100

Abstract
Because bug reports (BRs), which contain comprehensive information such as the description, status, a reporter, assignee, priority, severity, and other details about the bug, are a valuable tool for fixing defects found during software testing, researchers have focused their attention on the routine upkeep of software development systems. Since there are a lot more BRs in this process than there are problems in the system, the primary challenge is analyzing these BRs to find every issue. This is a laborious and time-consuming effort to conduct by hand. Thus, the optimal option is the automated one. Much of the current research concentrates on automating this procedure from many angles, such figuring out how serious or urgent the defect is. They did not, however, take into account the fact that the defect is a multi-class categorization issue. This research provides a solution to this issue by putting forward a novel prediction model to examine BRs and forecast the kind of bug. The suggested model combines machine learning and NLP (natural language processing) methods to build a combined machine learning algorithm. Using a publicly accessible dataset for three online software bug repository (Mozilla and Eclipse), that comprises six types (Program Anomaly, GUI, Networking or Security, Configuration, Efficiency, and Test-Code), we mimic the suggested model. The results of the simulation demonstrate that the suggested model can outperform the majority of current models in terms of accuracy, achieving 90.42% without text supplementation and 96.72% with it.

Keywords: Bug reports (BRs), software testing, defect fixing

Introduction
Testing, as used in software engineering, is the assessment procedure carried out to determine if a particular system satisfies the requirements. It involves identifying any errors or shortcomings in satisfying the criteria as established by the stakeholders. This procedure leads to the maintenance phase fixing flaws found after the testing phase ends. Additionally, software companies tend to provide defective software, and the likelihood of errors in software projects increases with program complexity and scale. Users report these found flaws and problems as a result. A software defect is an error, fault, failure, or flaw that causes the program to perform improperly or provide inaccurate results. The reporter submits a bug report, or feedback, to the bug tracker (BTS). shows an instance of a bug complaint from the Eclipse source.1. A bug report includes details about the found problem, including the issue's ID, its status (closed or opened), description, impacted component, program information, reproducibility instructions, bug reporter, and developer. You may think of a bug report as the channel via which the bug is sent to the developers. The bug management method is what the developer does to fix an issue when it is assigned to them. When consumers find a defect in a software product that has been published, they report the issue to the issue management program, which initiates the process. Developers are then tasked with investigating the precise spot of the issue after receiving this bug report. The developer who locates the problem's source and fixes it before other developers does so fixes the bug. Following the problem's resolution, the tester verifies the bug scenario and upgrades the bug report state to Verified if the issue does not recur. The reporter eventually gets a notice. Every software bug has a unique life cycle consisting of many stages. shows the life cycle of a bug report as a result. The three stages of the life cycle-bug management, bug evaluation, and bug localization-are shown in this image. Every action that takes place from the moment users report an issue until the developer assigns it is included in the bug management phase. Bug triage is the following stage, when reports have priority and matched with an appropriate developer to be fixed. The bug status is finally changed from solved to confirmed to closed during the localization step.

The primary issue in this stage of life is the exponential growth of reports of bugs, which are laborious, time-consuming, and difficult to handle manually. In order to overcome this problem, researchers examine the reports, divide them into three primary categories, each of which includes subcategories, and extract pertinent data that will speed up and simplify the maintenance process. The type, importance, and severity of these bug report categories are categorized. Most studies categorize bug reports according to priority or severity. The research's literature assessment demonstrates that different classification methods categorize problem reports according to different factors, but high-accuracy nature-based issue classification models are lacking. By presenting a combined machine learning technique for nature-based issue prediction from bug reports, this work aims to close this gap. The goal of this study is to contribute to this environment and provide a solution for software system maintenance by using text mining, machine learning, and NLP (natural language processing) approaches to automatically forecast the sorts of bugs. Rather of doing this laborious work manually, the model uses bug localization methods to speed up the maintenance phase after it has determined the kind of issue. shows a summary of how a bug report is used to forecast bugs. The suggested technique uses multiple machine learning (ML) initial classifiers and trains them on a benchmark dataset in an effort to improve nature-based bug prediction. In addition, we present ensemble neural network classifiers to enhance the accuracy of the model that is suggested by using both soft and hard voting. To increase the accuracy even further, we employ a text enhancement approach. The following are this paper's primary contributions:

It offers an assessment of a few foundational machine learning techniques for an automated categorization of bug reports based on their natural habitat.

- It presents a text augmentation strategy in nature-based bug predictions from problem reports in the suggested model.
- It suggests an automated ensembles machine learning oriented approach for bug predictions from bug reports. To the best of our understanding, this is the first ensembles machine learning method that uses text augmentation to predict the kind of defects found in bug reports.
- The suggested ensemble machine learning strategy seems to be effective in nature-based bug predictions from bug reports, according to the assessment findings obtained on a benchmark dataset.

Related Work

‘‘Software testing techniques: A literature review,’’

The complexity of today's computer programs combined with the urge to compete has driven quality assurance of produced software to unprecedented levels. Software evaluation is an essential component of the lifecycle of software development and should be handled with improved and effective methods and techniques due to its crucial role in the pre- and post-development phases of the process. In order to increase quality assurance, this article will address both current and upgraded testing methodologies.

‘‘Deep neural network-based severity prediction of bug reports,’’: The phase of creating software that involves software maintenance is crucial. Systems for tracking defects are used by developers to gather information for better software. Through these issue tracking systems, users report defects and determine the level of severity. An essential factor in determining how soon a defect has to be fixed is its severity. It assists developers in promptly resolving critical problems. However, determining severity manually is a laborious task that may be inaccurate. In order to do this, we provide an automated method for predicting the severity of bug reports based on deep neural networks in this study. Initially, we use natural language processing methods to preprocess bug reports' content. Second, we give every bug report an emotion score that we calculate and assign. Third, we give every preprocessed bug report a vector. Next, in order to estimate the severity of each bug report, we feed the created vector plus the emotion score to a classifier that uses deep neural networks. Additionally, we assess the suggested methodology using bug report history data. According to the cross-product data, the suggested strategy performs better than the most advanced methods. It raises the f-measure by an average of 7.90%.

‘‘A method of non-bug report identification from bug report repository,’’

A software program called a bug tracker (BTS) is created to assist software development teams in monitoring software defects that are reported by testers and end users. Nonetheless, some bug reports that end users have sent to the BTS may not really be bugs. Generally, non-bug reports submitted to the BTS system are manually screened and filtered out by bug triagers, who are experts in software development. This work requires a lot of effort and time. This presents a problem for the research, whose goal is to offer a way of classifying bug reports using a one-class support vector machines (SVM). Three aspects of bug reports and three weighting algorithms (tf, tf-idf, and tf-igm) are examined in order to determine the best classifiers for bug reports. When compared to earlier studies in this field, the experiment's outcomes can be deemed satisfactory.

‘‘Automated bug classification.: Bug report routing,’’

We focus on the issue of categorizing complaints of software bugs. Building a classifier that can categorize freshly received bug reports into the corrective (defect repair) report and flawless (major maintenance) report-two established classes-is our primary goal. This enables maintainers to assign resources for every group and rapidly comprehend these issue reports. We suggest a unique feature set based on the frequency of certain terms for this reason. A variety of classification techniques are then used to create a classification model using the suggested feature set. With a mean accuracy of 93.1% across three distinct open-source projects, the results of the suggested feature set's categorization using the SVM classification method demonstrated great accuracy.

Methodology

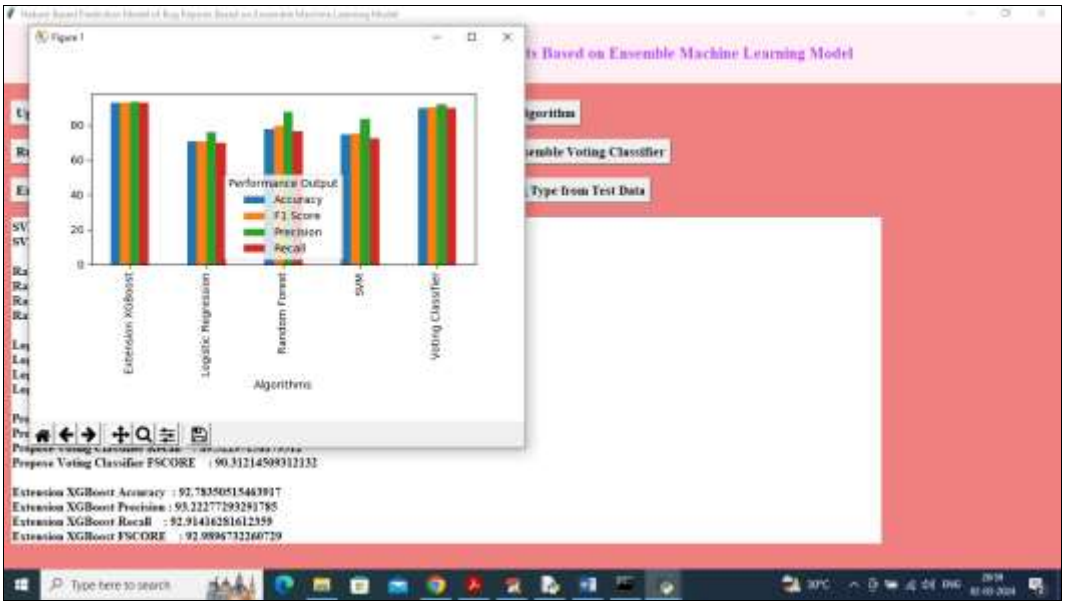
To implement this project we have designed following modules

1. **Upload Eclipse Mozilla Bug Dataset:** This module allows users to submit a dataset to an application, clean all text data, and then remove stop words, special symbols, lemmatization, and stemming.
2. **Process Dataset:** should transform the whole text data

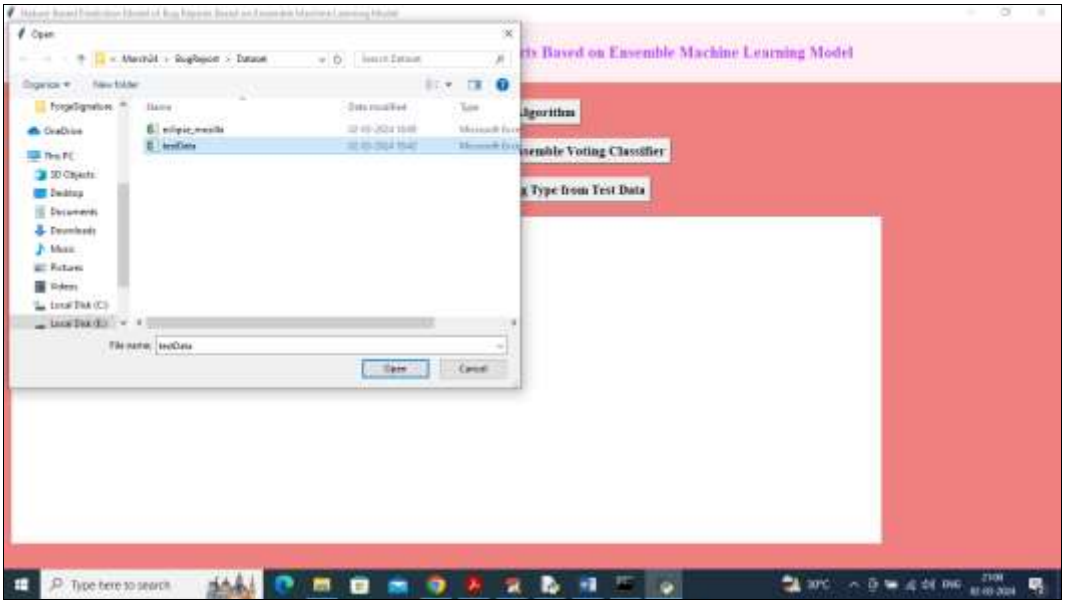
into an enhanced numerical TF-IDF vector, replacing each word with its mean frequency. TF-IDF will undergo shuffling and normalization. The program will use 80% of the processed data for training & 20% for testing once the data has been separated into training and evaluation categories.

3. **Run SVM Algorithm:** To create a model, 80% of the training data is going to be fed into the SVM algorithm. The model will then be applied to 20% of the test data to determine metrics like accuracy.
4. **Run Random Forest Algorithm:** The Random Forest technique will use 80% of the training data to create a model, which will then be applied to 20% of the test data to determine accuracy and other metrics.
5. **Run Logistic Regression Algorithm:** To create a model, 80% of the training data is going to be fed into the LR algorithm. The model will then be applied to 20% of the test data to determine metrics like accuracy.
6. **Propose Ensemble Voting Classifier**: To create a model, the Voting Classifier algorithm will get 80% of the training data. The model will then be applied to 20% of the test data to determine accuracy and other metrics.
7. **Extension XGBoost Algorithm:** To create a model, 80% of the data used for training will be fed into the XGBoost algorithm. The model will then be applied to 20% of the test data to determine metrics like accuracy.
8. **Comparison Graph:** will provide a graph that compares all of the algorithms.
9. **Predict Bug Type from Test Data:** The system will identify the sort of issue by uploading test data using this module.

Results and Discussion

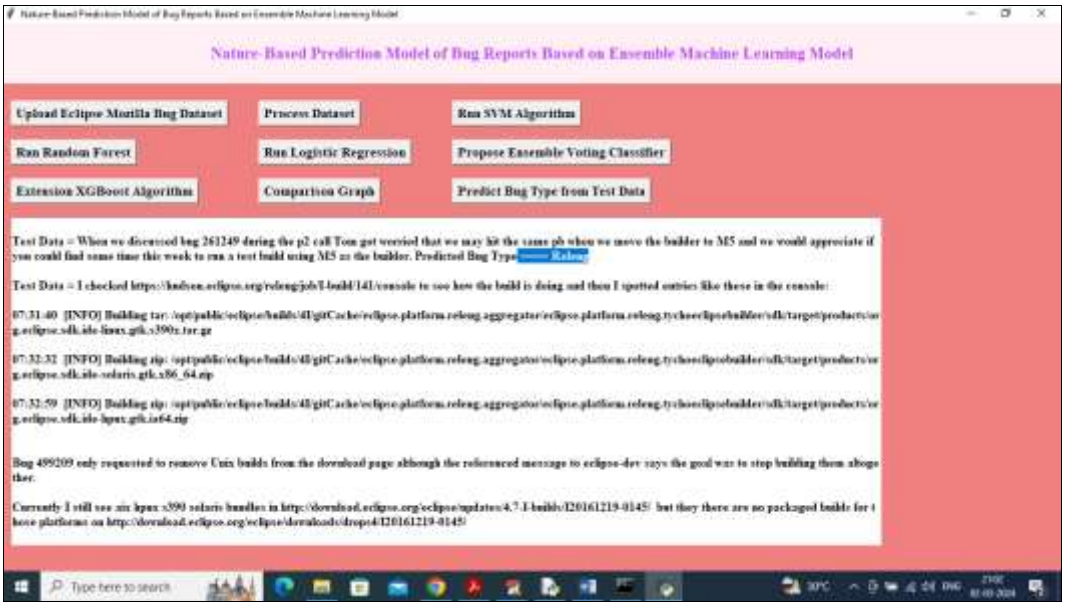


In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms Extension got high accuracy and now click on 'Predict Bug Type from Test Data' button to upload test data and get below output.



In above screen selecting and uploading test data file and then click on 'Open' button to get below output.

In above screen can see all text data from BUG and the after arrow symbol =➔ can see predicted Bug Type and below is another sample.



Conclusion

Using a combination machine learning technique made up of four basic machine learning algorithms-Random Forest, Support Vector Machine Classification, logarithmic regression, & Multinomial Naïve Bayes-this work presented a nature-based bug detection component. The model has a 90.42% accuracy rate. In addition, it makes use of a text augmentation method to improve accuracy. Consequently, the suggested model's maximum accuracy reached 96.72%. Program anomaly, GUI, Networking or Safety, Configuration, Efficiency, and Test-Code are the six problem categories from which the suggested model predicts the kind of issue. In order to decrease the maintenance time, future work will improve this model by adding more issue categories and suggesting potential fixes for anticipated bugs.

References

1. Jamil MA, Arif M, Abubakar NSA, Ahmad A. Software testing techniques: A literature review. In: Proceedings of the 6th International Conference on Information and Communication Technology in the Muslim World (ICT4M); Nov 2016. p. 177-182.
2. Ramay WY, Umer Q, Yin XC, Zhu C, Illahi I. Deep neural network-based severity prediction of bug reports. IEEE Access. 2019;7:46846-46857.
3. Wen W. Using natural language processing and machine learning techniques to characterize configuration bug reports: A study. M.S. thesis, College of Engineering, University of Kentucky, Lexington, KY, USA; C2017.
4. Polpinij J. A method of non-bug report identification from bug report repository. Artif Life Robot. 2021;26(3):318-328.
5. Adhikarla S. Automated bug classification: Bug report routing. M.S. thesis, Faculty of Arts and Sciences, Department of Computer and Information Science, Linköping University, Linköping, Sweden; c2020.
6. Youm KC, Ahn J, Lee E. Improved bug localization based on code change histories and bug reports. Inf

Softw Technol. 2017;82:177-192.
7. Safdari N, Alrubaye H, Aljedaani W, Baez BB, DiStasi A, Mkaouer MW. Learning to rank faulty source files for dependent bug reports. In: Proceedings of SPIE; 2019;10989;109890B.
8. Kukkar A, Mohana R, Nayyar A, Kim J, Kang B-G, Chilamkurti N, *et al*. A novel deep-learning-based bug severity classification technique using convolutional neural networks and random forest with boosting. Sensors. 2019;19(13):2964.
9. Aggarwal A. Types of bugs in software testing: 3 classifications with examples; c2020 May [cited 2024 Sept 22]. Available from: https://www.scnsoft.com/software-testing/types-of-bugs
10. Kukkar A, Mohana R. A supervised bug report classification with incorporate and textual field knowledge. Proc Comput Sci. 2018;132:352-361.
11. Otoom AF, Al-jdaeh S, Hammad M. Automated classification of software bug reports. In: Proceedings of the 9th International Conference on Information and Communication Management; c2019 Aug. p. 17-21.
12. Morrison PJ, Pandita R, Xiao X, Chillarege R, Williams L. Are vulnerabilities discovered and resolved like other defects? Empirical Softw. Eng. 2018;23(3):1383-1421.
13. Lopes F, Agnelo J, Teixeira CA, Laranjeiro N, Bernardino J. Automating orthogonal defect classification using machine learning algorithms. Future Gener Comput Syst. 2020;102:932-947.
14. Hirsch T, Hofer B. Root cause prediction based on bug reports. In: Proceedings of the IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW); c2020 Oct. p. 171-176.
15. Umer Q, Liu H, Illahi I. CNN-based automatic prioritization of bug reports. IEEE Trans Rel. 2020;69(4):1341-1354.