J. Sci. Trans. Environ. Technov. 2023, 17 (1) : 46 - 52

A REVIEW ON BUG REPORT CLASSFICATION USING MACHINE LEARNING APPROACHES

P Janarthanan[1*], S. Sreeram[2] , S. Sherlin Shibi[3], R. Shakthi[4]

[1] Professor, Department of Computer Science and Engineering Sri Venkateswara College of Engineering Sriperumbudur,Kanchipuram, TamilNadu, India. janap@svce.ac.in
[2,3,4] Department of Computer Science and Engineering Sri Venkateswara College of Engineering Sriperumbudur, Kanchipuram, TamilNadu, India.
janap@svce.ac.in ,sreeram2k03@gmail.com, shibi2114@gmail.com,shakthiravi810@gmail.com

Article History
Received: **12.07.2023**
Revised and Accepted: **10.08.2023**
Published: **15.09.2023**

https://doi.org/10.56343/STET.116.017.001.007
www.stetjournals.com

ABSTRACT

Bug reports are crucial documents in software development, providing detailed information about software issues, including descriptions, current status, and how severe they are. They are essential for identifying and keeping track of software problems, which are essential for ensuring the overall quality of software systems. When there are no unresolved bugs, it's a clear indicator that the software is reliable and works smoothly. In recent years, machine learning has becomeincreasingly skilled at classifying different types of software bugs. One way it does this is by using ensemble machine learning models that combine various tools like random forests, decision trees, and naive Bayes. Additionally, support vector machines (SVM), decision trees (C4.5), and other classification methods have been used to better understand and categorize bugs. This paper focuses on how machine learning can revolutionize bug tracking, nature of bug, contributing to the ongoing conversation about making software more dependable.

Keywords: Bug Reports, Machine Learning, Nature of Bug, Bug Tracking and Ensemble Machine Learning.

P Janarthanan

Department of Computer Science and Engineering Sri Venkateswara College of Engineering Sriperumbudur, Kanchipuram, TamilNadu, India.

email : janap@svce.ac.in

P-ISSN 0973-9157
E-ISSN 2393-9249

INTRODUCTION

Software bugs are common issues that can impact the functionality and reliability of software applications (Jamil *et al.*, 2016). Timely identification and management of these bugs are crucial for ensuring the quality of software products (Wen, 2017). To address this challenge, machine learning techniques have been increasingly applied to predict and manage software bugs (Ramay *et al.*, 2019). In this context, a Nature-Based Prediction Model of Bug Reports based on an Ensemble Machine Learning Model represents a sophisticated approach to improve the accuracy and effectiveness of bug

prediction (Polpinij, 2021). Common types of software bugs are functional Bugs, user interface (UI) Bugs, compatibility issues, and performance issues (Adhikarla, 2020).

The nature-based prediction model of bug reports based on ensemble machine learning model isa new approach to automatically classify bug reports into different categories such as configuration, network or security, GUI, Program Anomaly, Performance, Test Code (Safdari *et al.*, 2019). F1 score for each class is shown in fig.1 It uses a combination of natural language processing (NLP) and machine learning techniques to extract features from the bug reports and then classify them using an ensemble of machine learning models (Kukkar et al., 2019). Refer Figure 1 for F1 scores of Bug classes.

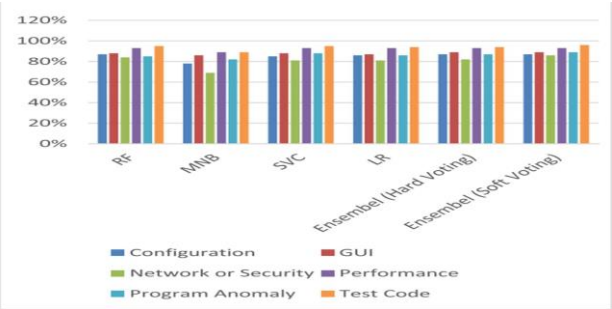


Figure.1 F1 scores for Bug classes

The NLP techniques are used to extract features from the text of the bug reports, such as the words and phrases used, the sentiment of the text, and the grammatical structures. The machine learning models are then used to classify the bug reports based on these features. The ensemble machine learning model is a combination of different machine learning models, such as decision trees, support vector machines, and random forests.

This combination of models helps to improve the accuracy of the classification. The nature-based prediction model of bug reports has been shown to be effective in classifying bug reports into different categories. In a study, the model was able to achieve an accuracy of 90.42% without text augmentation and 96.72% with text augmentation.

The nature-based prediction model of bug reports is a promising new approach to automatically classify bug reports. It can help to reduce the time and effort required to manually classify bug reports, and it can also help to improve the accuracy of the classification.

The nature-based prediction model of bug reports offers a multitude of advantages to software development and testing processes. One of its key benefits lies in its ability to significantly reduce the time and effort needed for the manual classification of bug reports. This streamlined classification process not only saves valuable resources but also enhances the accuracy of bug categorization, ensuring that issues are properly identified and addressed. Furthermore, this model's capacity to recognize patterns within bug reports holds immense potential for improving the overall software development process. By discerning recurring issues or common threads among reported bugs, developers can proactively address underlying problems and enhance the software's overall quality.

This predictive capability also aids in prioritizing bug reports, enabling teams to focus their attention on critical issues that require immediate resolution. In essence, the nature-based prediction model for bug reports stands as a valuable tool for both software developers and testers.Its implementation leads to quicker and more efficient bug identification and resolution, ultimately resulting in a higher-quality software product. This is exemplified in Figure 2, which showcases an illustrative example of a bug report.

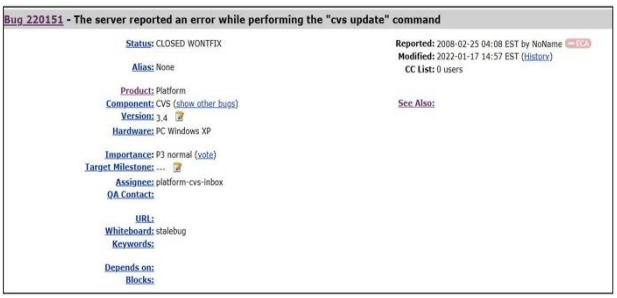


Figure.2. Example of Bug

MACHINE LEARNING TECHNIQUES

ML is sub-area of Artificial Intelligence and has the ability of a machine to learn by themselves with a large number of datasets. It uses the statistical and mathematical approach to solve the problem. Currently, the areas where neural networks are Signal Processing, Pattern Recognition, Medicine, Speech Production and Speech Recognition. ML enables the system to recognizepatterns based on present and past data. Every dataset is described with several features. Each sample is pre-processed to remove the unnecessary noise, outliers and there will be chances ofdata missing which can decrease the quality of resultant prediction. Then, slice the dataset is classified into two phases (1) Training phase, (2) Testing phase. In the training phase, the data is split in the ratio 80:20. A major part of the data is made to train the system to result in greater accuracy. In the testing phase, the system is tested based on trained data. The testing must meet the following conditions Large enough to yield statistically meaningful results.ML algorithms are classified into Supervised Learning, Unsupervised Learning. Supervised learning involves what has been learned in the past using labelled samples to predict the future. A supervised learning algorithm analyses the training data and produces inferred function which can be usedfor mapping new samples. The Naïve Bayes classifier is one of the supervised learning approaches which are a probabilistic machine learning technique for classification purpose. It assumes the presence of a particular feature in a class is unrelated to the presence of any other feature. Classification can be performed using eqn.

$$P(A \mid B) = P(A \mid B) * P(A) / p(B) \qquad -(1)$$

Where A and B are events and P(A) expressed as Marginal likelihood, P(B) expressed as prior Probability, P(A|B) denotes Likelihood. It is scalable and a great choice for real-world applications. Fig.3 explains the classification between Age and Estimated salary. The outliers are removed to make a better prediction. In our case, age is the independent variable and estimated salary is considered as the dependent variable. The GaussianNB function is invoked to classify the given data. The other supervised algorithms are Decision Tree (DT), Polynomial Regression, Random Forest, Support Vector Machine etc.; from the feature extracted from the image it is easy to classify the presence of cancer using Support Vector Machine (SVM) [15]. Classification between Age and Salary is shown fig.3.

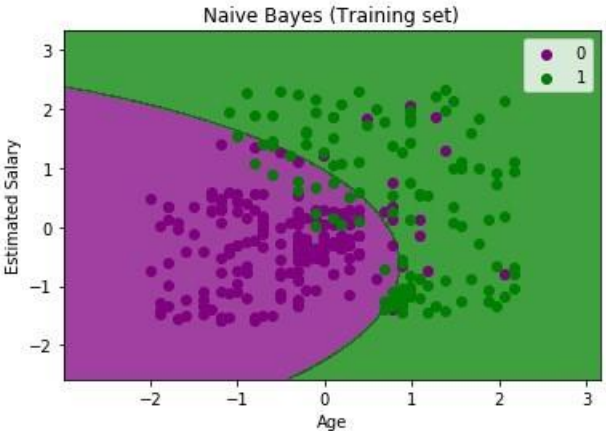


Figure 3. Classification between Age and Salary

ALGORITHM

Input: Dataset N after Binary Conversion.
Output: Presence of Tumor Cells

1) Reading the training dataset N.
2) Calculate the Dimensionality and Geometry of the particular Organ.
3) Repeat until the probability of the function using Gauss density equation in each class until theprobability of all predicator value.
4) Calculate the likelihood of each class.
5) Get the greatest likelihood

Unsupervised learning is the task of machines to learn using data sets with no specified structure. Two methods used in unsupervised learning such as Principal Component Analysis and Clustering. Cluster is also the sub-area of ML which can cluster the data to be classified. Clustering is a powerful technique used to organize the data based on similarity in the dataset. The unsupervised learning algorithms include K-Means clustering, Hierarchical clustering etc. Though unsupervised learning helps to identify the patterns, the ability to extract the compressed representation accurately may still need to be tested to determine the appropriateness of the implementation (Wen, 2017). In fig.3 the left diagram represents how the data points are plotted and the other explains how

the plotted data points are clustered into 3 major categories. Example of clustering is shown in fig.4.

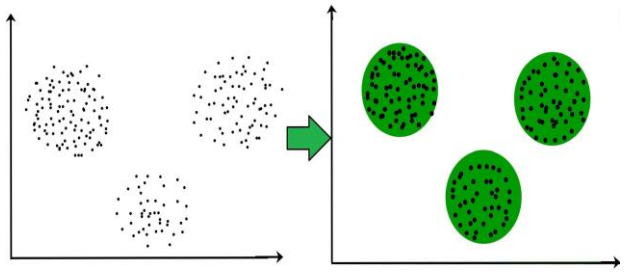


Figure 4. Clustering

MACHINE LEARNING APPLICATIONS

Machine Learning (ML) applications on bug reports have gained prominence in software development and quality assurance processes. ML techniques are leveraged to streamline bug management, prioritize bug fixes, and enhance overall software quality.

SEVERITY PREDICTION OF BUG REPORTS

Developers employ issue tracking systems to collect bugs for software enhancement. druggies submit bugs through similar issue tracking systems and decide the inflexibility of reported bugs. The inflexibility of a bug is an important trait that determines how snappily it should be answered. It helps inventors break important bugs on time. still, homemade inflexibility assessment is a tedious job and could be incorrect. To this end, in this paper, we propose a deep neural network- grounded automatic approach for the inflexibility vaticination of bug reports. First, we apply natural language processing ways for the textbook preprocessing of bug reports. Second, we cipher and assign an emotion score for each bug report. Third, we produce a vector for each pre- processed bug report. Fourth, we pass the constructed vector and the emotion score of each bug report to a deep neural network- grounded classifier for inflexibility vaticination. An overview of the deep neural network-based severity prediction of bug reports is presented in the below Figure 5.

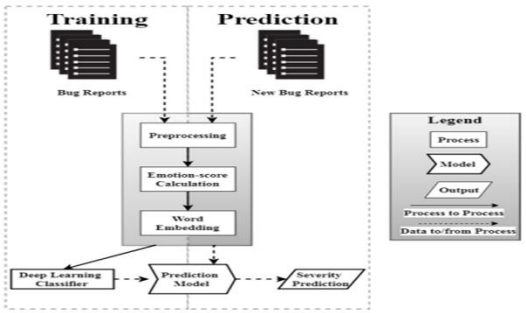
Figure. 5. Severity Prediction

We also estimate the proposed approach based on the history of bug reports. The results of the cross-product analysis suggest that the proposed approach outperforms state-of-the-art approaches. On average, it improves the f-measure by 7.90.

The proposed approach predicts the inflexibility of bug reports as follows: First, we value the history data of bug reports from open-source systems. Second, we preprocess the bug reports using natural language processing methods. Third, we cipher and assign an emotion score to each bug report. Fourth, we produce a vector (word embeddings) for each pre-processed bug report. Eventually, we train a deep literacy-rooted classifier for inflexibility vaticination. We input the emotion score and the vector of each bug report to the classifier for its inflexibility.

We use the given illustration to illustrate how the proposed approach predicts the inflexibility of bug reports. It's a decline bug report(# 437094) collected from Bugzilla(2). Product = '' ECP '' is the name of the affected product. Textual Information The EMFFilter must be streamlined to sludge out new models added to Luna, is the description of the bug report. It may contain details on how the bug can be regenerated. Inflexibility = ''critical '' is the inflexibility of the bug report that indicates how snappily a bug report should be resolved.

To answer the exploration question RQ1, we compare the proposed approach against EWD-Multinomial and prognosticate in the inflexibility validation of bug reports. The cross- project confirmation results of the proposed approach, EWD-multinomial, and prognostic are presented. For each table, the first column presents the products. Columns 2–6 present the accuracy, perfection, recall, f-measure, and MCC of each approach (Table 1).



Product	Accuracy	Precision	Recall	F-measure	MCC
CDT	87.81%	80.12%	83.81%	81.92%	0.224
JDT	88.71%	83.89%	88.71%	86.23%	0.398
Platform	88.34%	84.18%	88.35%	86.21%	0.235
Core	89.70%	86.39%	89.70%	88.01%	0.266
Firefox	83.23%	81.61%	83.46%	82.52%	0.33
Thunderbird	89.72%	79.68%	81.94%	80.79%	0.313
Bugzilla	89.18%	82.60%	87.18%	84.83%	0.234
Average	**88.10%**	**82.64%**	**86.16%**	**84.36%**	**0.286**

Table 1 Severity Prediction Report

DUPLICATE BUG REPORT DETECTION AND CLASSIFICATION SYSTEM

Duplicate bug report detection and classification systems are essential components of efficient software development and bug tracking processes. These systems are specifically designed to automatically identify and categorize bug reports that are duplicates, meaning they describe the same or very similar issues in a software application. Their primary objectives encompass several key aspects:

Firstly, they focus on identifying duplicates, employing algorithms that analyse the textual content of bug reports, including their titles, descriptions, and any attached files or logs. This step is crucial in eliminating redundancy and ensuring that developers do not spend unnecessary time addressing the same issue repeatedly. Secondly, these systems aim to reduce redundancy by flagging or merging duplicate bug reports. This not only saves valuable developer time butalso streamlines the bug tracking database, making it more organized and efficient to manage. Furthermore, they facilitate categorization of duplicate bug reports, assisting developmentteams in prioritizing and addressing issues based on their severity, priority, or other relevantcriteria. This ensures that the most critical problems are dealt with promptly. Additionally, some systems can also automate user notification regarding duplicate reports, notifying users whosubmitted them about the duplication and providing transparency in the bug resolution process.These systems are not only tools for bug management but also valuable sources of data analysis.By analysing patterns of duplication, development teams can gain insights into recurringproblems or areas of the software that require special attention, ultimately contributing to software quality improvement. Moreover, many of these systems incorporate advancedtechnologies like machine learning and natural language processing (NLP) to continuously enhance their accuracy. They learn from historical data, improving their ability to recognize duplicate reports over time. In summary, duplicate bug report detection and classification systems serve as indispensable assets in bug tracking, streamlining communication between users and developers, and ultimately elevating the quality and reliability of software products. The Figure 3.2 denotes Duplicate Bug Report Detection and Classification System.

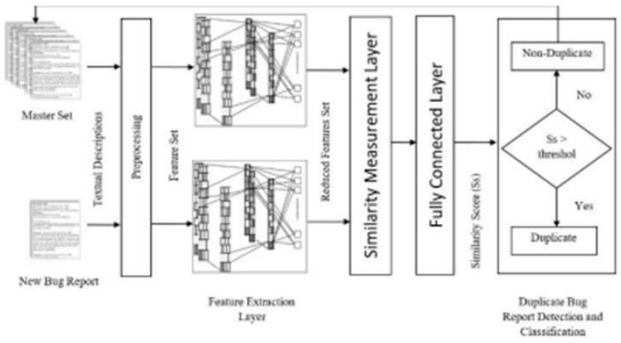


Figure 6. Duplicate Bug Report Detection and Classification System

The proposed system comprises of three modules: Preprocessing: The Figure 6 denotes Duplicate Bug Report Detection and Classification System. The basic aim of this module is to convert each term of the bug report into more manageable representation, remove the unwanted terms from the bug reports. Deep Learning Model: In this module deep learning-based model is proposed for extracting the semantic and syntactic relationship of words for the textual similarity measurement between bug reports. CNN based Feature Extraction Layer: CNN technique is usedto extract the relevant features. It has various convolution filters that capture the local featuresand examines all words of the bug reports from multiple perspective. Similarity Measurement Layer: This layer contains similarity measurement metric, which compare the sentence representation of the bug reports. Fully Connected Layer: This layer computes the similarity score of the sentences of bug reports. Duplicate Bug report Detection and Classification: This module classifies the duplicate bug report from non-duplicate bug report based on the final decision based on the similarity scores.

The Figure 7 explains the Flowchart for Duplicate Bug Report Detection and Classification System.

The experimental results of proposed system are demonstrated in this subsection. To predict the bug is duplicate or not, the proposed system based on deep learning model is used. The six datasetsand five performance metrics are adopted to evaluate a performance of duplicate bug report detection and classification system as mentioned above. The proposed system adopted the binary classification schema. It can be seen that proposed system that proposed system effectively computes the accuracy, precision, recall and f-measure rates for each duplicate and non-duplicate bug report of each dataset. It can be seen from table, precision of proposed system for NetBeans, Eclipse, Open office, Gnome, Mozilla, Combined and Firefox datasets is 82.90%, 97.23%, 97.44%, 86.16%, 98.70%, 94.35% and 8024% respectively. The recall rate of NetBeans, Mozilla, Eclipse, Open office, Gnome, Combined and Firefox datasets obtained by proposed system is 80.23%, 98.42%, 97.04%, 95.34%, and 83.35% 96.34% and 81.23% respectively.

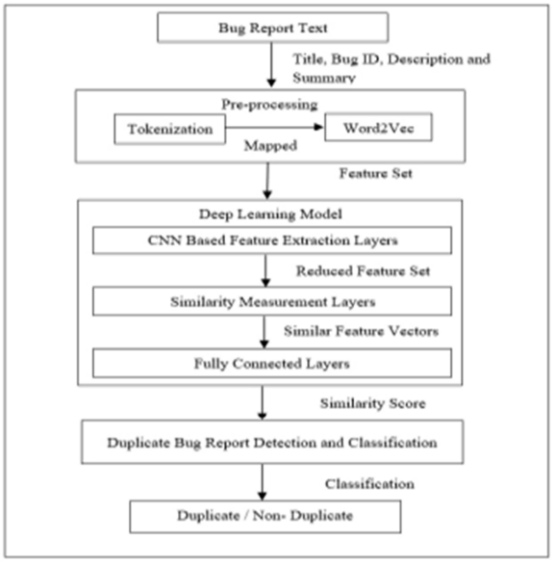


Figure 7 Flowchart for Duplicate Bug Report Detection and Classification System

CONCLUSION

This paper proposed a nature-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 accuracyof 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

Adhikarla S., 2020, "Automated bug classification. Bug report routing,"M.S. thesis, Fac. Arts Sci., Dept. Comput. Inf. Sci., Linköping Univ., Linköping", Sweden.

Aggarwal. A, 2020, Types of Bugs in Software Testing Classifications with Examples".

Jamil M. A., Arif M., Abubakar N. S. A., and Ahmad A., 2016, '', Software testing techniques: A literature review''. in Proc. 6th *Int. Conf. Inf. Commun. Technol.* Muslim World (ICT4M), 177-182, https://doi.org/10.1109/ICT4M.2016.045

Kukkar A., Mohana R., Nayyar A., Kim J., Kang B.G., and Chilamkurti N., 2019, "A novel deep- learning-based bug severity classification technique using convolutional neural networks and random forest with boosting", *Sensors*,19(13), pp. 2964. https://doi.org/10.3390/s19132964 PMid:31284398 PMCid:PMC6651582

Kukkar A. and Mohana R., 2018, "A supervised bug report classification with Incorporate and Textual Field Knowledge", *Proc. Comput. Sci*., vol. 132, pp. 352-361. https://doi.org/10.1016/j.procs.2018.05.194

Morrison P. J., Pandita R., Xiao X., Chillarege R., and Williams L., 2018, "Are vulnerabilities discovered and resolved like other defects?, Empirical Software Eng.", 23(3), pp. 1383- 1421. https://doi.org/10.1007/s10664-017-9541-1

Otoom A. F., Al-jdaeh S., and Hammad M., 2019, "Automated classification of software bug reports",in Proc. 9th *Int. Conf. Inf. Commun. Manage*., pp. 17-21. https://doi.org/10.1145/3357419.3357424

Polpinij, 2021, "A method of non-bug report identification from bug report repository", Artif. Life Robot. 26(3), pp. 318-328. https://doi.org/10.1007/s10015-021-00681-3

Ramay W. Y., Umer Q., Yin X. C., Zhu C., and Illahi I., 2019, "Deep neural network-based severity prediction of bug reports", IEEE Access, 7, pp. 46846-46857. https://doi.org/10.1109/ACCESS.2019.2909746

Safdari N., Alrubaye H., Aljedaani W., Baez B. B., DiStasi A., and Mkaouer M. W., 2019, "Learning to rank faulty source files for dependent bug reports", in Proc. SPIE, 10989. https://doi.org/10.1117/12.2519226

Wen W., 2017. "Using natural language processing and machine learning techniques to characterize configuration bug reports", M.S. thesis, College Eng., Univ. Kentucky, Lexington, KY, USA.

Youm K. C., Ahn J., and Lee E., 2017, "Improved bug localization based on code change histories and bug reports", *Inf. Softw. Technol.*, 82, pp. 177-192, 2017. https://doi.org/10.1016/j.infsof.2016.11.002