Predicting Software Defect Complexity Using Bug Tracking

[1]Korrapati Bhuvaneswara Chari, [2]Verpula Yashwanth Kumar, [3]Reddypally Saikar, [4]B Savit

[1,2,3,4]Students
Department of Computer Science and Engineering,
Bharath Institute of Higher Education and Research, Chennai, India

***Abstract*--Numerous free, open-source, and paid bug tracking programmes have been created or are constantly being created. As there are more problems with software projects every day, developers are beginning to use bug tracking tools to keep track of the bug reports. The industry needs the criteria for choosing the best system tool from the assortment of system tools that are currently accessible in order to track and correct bugs and report them as they are fixed. While there are numerous bug tracking solutions that deliver the data via different resources, such as web interfaces, it is still a challenge to gather usable information from the huge and disorganised set of complaints. To handle the reviews of the available tools and offer a new improved tool for the bug tracking and reporting system, we attempt to present these thorough classification criteria. Assigning the bug to the developer for monitoring and resolving the progress of bug fixing through various graphical/charting facilities and status updates also helps in reporting the bugs that are discovered by that method. Additionally, it attempts to detect bugs for complexity measures and offers the reliability of bug prediction, enabling distribution of patches to consumers.**

***Index Terms*— Bug, Support Vector Machine, Convolutional Neural Network, SMOTE.**

I.INTRODUCTON

This Bug Tracking System project's primary goal is to deal with offering online support to software engineers who are encountering bugs or faults in software technologies. Project details, developer details, and tester details can all be maintained by this project. The system that makes it possible to find bugs is called a bug tracking system. Although it doesn't find the bugs, it does give detailed information about those that have been found. A user of a bug tracking system is able to give information regarding bugs that have been found. Engineers create the project in accordance with client specifications. During the testing process, the tester will locate the bugs.

When a tester encounters a large number of issues, he adds the bug id and details to the database. The project manager and developer are informed by the tester. Project managers and developers have access to the database table's bug details. Whenever a customer places an order or a request for a product to be developed. The task of adding users to the bug tracking system and allocating projects to the users falls under the purview of the system/project manager. This project offers information on bugs, such as bug id, name, priority, project name, location, and kind. Until all of the problems in this system are fixed, all processes are running continually. As soon as a bug is found, this system can assist in reporting it to the project manager, system manager, and developer. Anyone who actually needs to know about the defect can easily learn about it shortly after it is reported. The testing phase is where the bug tracking system is most important. However, it allows for the assignment of projects to developers and testers. This bug tracking system keeps track of various users and offers distinct workspaces for project managers, developers, and testers.

II.METHODOLOGY

1. An Eye Tracking Research on Debugging Strategies towards Different Types of Bugs , FeiPengFeng, et.al IEEE 2021.

Debugging is one of the important links in software quality assurance. Generally, the debugging performance of people adopting different debugging strategies varies enormously. Although there are studies discussing debugging strategies, only a small amount of research examines how these tactics affect various bug types. The trials done on 20 participants in this paper's experiments reveal that there are differences between the eye movement data of those successful and unsuccessful debugging samples. Paying attention to variable changes is helpful when dealing with data flow errors, but when handling control flow bugs, it's more crucial to watch the code and comprehend their logical structure. Combining this conclusion with the compiler's error message, in our opinion, can aid programmers in more effectively locating flaws.

2. The Eclipse and Mozilla defect tracking dataset: A genuine dataset for mining bug information Ahmed Lamkanfi; et.al IEEE 2021

The analysis of bug reports is an important subfield within the mining software repositories community. It explores the rich data available in defect tracking systems to uncover interesting and actionable information about the bug triaging process. While bug data is readily accessible from systems like Bugzilla and JIRA, a common database schema and a curated dataset could significantly enhance future research because it allows for easier replication. Consequently, in this paper we propose the Eclipse and Mozilla Defect Tracking Dataset, a representative database of bug data, filtered to contain only genuine defects (i.e., no feature requests) and designed to cover the whole bug-triage life cycle (i.e., store all intermediate actions). We have used this dataset ourselves for predicting bug severity, for studying bug-fixing time and for identifying erroneously assigned components. Sharing these data with

the rest of the community will allow for reproducibility, validation and comparison of the results obtained in bug-report analyses and experiments.

3.Feature Ranking and Aggregation for Bug Triaging in Open-Source Issue Tracking Systems AnjaliGoyal; et.al IEEE2021

The increasing complexity and team-based projects have led to the rise of various project management tools and techniques. One of the important components of open- source project management is the usage of bug tracking systems. In the last few decades, software projects have experienced an inescapable appearance of bug reports. One of the main challenges in handling these incoming bugs is triaging of bug reports. Bug triaging can be considered as a mechanism for the election of a suitable software developer for a reported bug who will work towards resolving bug in a timely fashion. There exist several semi and fully automated bug triaging techniques in the existing literature. These techniques often consider varied bug parameters for prominent developer selection. Past researchers have concluded different parameters to be possessing prime importance in the optimal developer selection task. However, a common ranking scale depicting the importance among different bug parameters for bug triaging is not available. This paper presents a methodology to rank the non-textual bug parameters using feature ranking and aggregation techniques. The presented methodology has been evaluated on four open-source systems, namely, Mozilla Firefox, Eclipse, GNome and Open Office. From the experimental evaluation, it has been observed that the ranking of bug parameters is consistent among the different open-source projects of Bugzilla repository.

4 A bug Mining tool to identify and analyze security bugs using Naive Bayes and TF-IDF DikshaBehl; et.al IEEE2021

Bug report contains a vital role during software development, However bug reports belongs to different categories such as performance, usability, security etc. This paper focuses on security bug and presents a bug mining system for the identification of security and non-security bugs using the term frequency-inverse document frequency (TF-IDF) weights and naïve bayes. We performed experiments on bug report repositories of bug tracking systems such as bugzilla and debugger. In the
proposed approach we apply text mining methodology and TF-IDF on the existing historic bug report database based on the bug s description to predict the nature of the bug and to train a statistical model for manually mislabeled bug reports present in the database. The tool helps in deciding the priorities of the incoming bugs depending on the category of the bugs i.e. whether it is a security bug report or a non-security bug report, using naïve bayes. Our evaluation shows that our tool using TF-IDF is giving better results than the naïve bayes method.

III.SUPPORT VECTOR MACHINE (SVM):

The primary function of the SVM is to find an ideal hyperplane for various distinct cases in a high dimensional space. Multiple hyperplanes exist to realise this paradigm. This process is dependent on the support vector, which is the data that corresponds to the ideal choice surface and is located closest to the closed surface. It carries out classification by generating a hyperplane to divide the data and planning the input vectors into a high dimensional space. This approach is mostly used to resolve non-convex, unconstrained minimization problems and quadratic programming problems. The SVM is the classifier process's most successful technique. Here , The SVM is a proposed System.

IV.CONVOLUTIONNEURAL NETWORKS (CNN):

A class of deep neural networks called convolutional neural networks (CNN, or ConvNet) are frequently used to analyse visual imagery. As a result of their shared-weights architecture and translation-invariance properties, they are often referred to as shift invariant or space invariant artificial neural networks (SIANN). They are used in a variety of fields, including image and video recognition, recommender systems, image classification, image segmentation, and medical image analysis. They are also used in brain-computer interfaces, natural language processing, and financial time series.

Multilayer perceptrons are modified into CNNs. Fully linked networks, or multilayer perceptrons, are those in which every neuron in one layer is connected to every neuron in the following layer. These networks are vulnerable to overfitting data because of their "fully-connectedness." Adding some kind of magnitude measurement of weights to the loss function is a typical method of regularisation.

CNNs tackle regularisation differently; they make use of the data's hierarchical pattern to piece together more complicated patterns out of smaller, simpler ones. CNNs are therefore at the lower end of the connectivity and complexity spectrum.

Because of how closely the connectivity pattern between neurons mirrors the structure of the animal visual cortex, convolutional networks were inspired by biological processes. Only in the constrained area of the visual field known as the receptive field do individual cortical neurons respond to inputs. Different neurons' receptive areas partially overlap one another to fill the whole visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

We have used CNN as a proposed system.

It has the following advantages:

- No manual.
- Data cannot be lost or destroyed
- It is easy to update, delete and view data.
- Maintaining and retrieving the record of Users is easy.
- Time consumption is low.
- Can be implemented in all datasets.

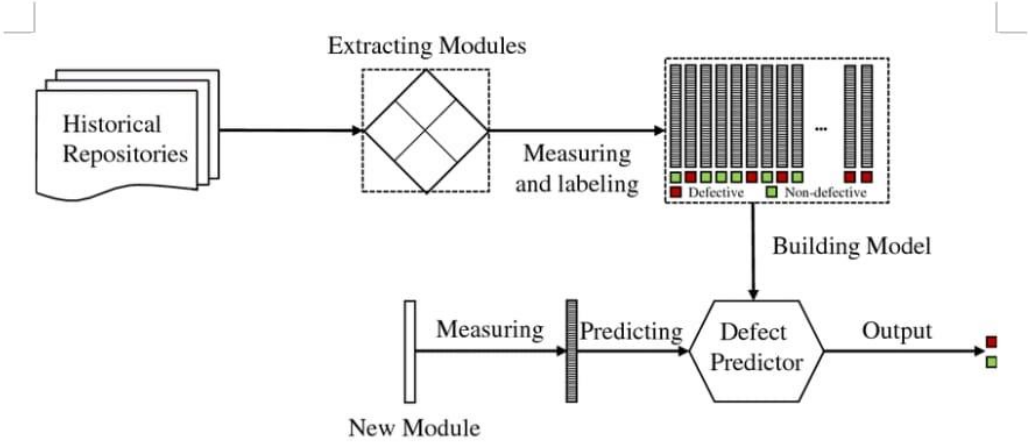

Fig 1 System Architecture

V.MODULE DESCRIPTION

1. Input dataset
2. Analysis of size of data set.
3. Oversampling.
4. Training and Testing.
5. Apply algorithms.
6. Predict results.

1. Input dataset:
Dataset can be taken from online data source provider from the internet sources. We have to collect a huge dataset in volume so as to predict the accuracy in an efficient manner.

2. Analysis of data set:
Here the analysis if dataset takes place. The size of data is taken into consideration for the data process.

3. Oversampling (Using SMOTE): we have created a detailed history of all Ebug that is been happened over a long duration. SMOTE (Synthetic Minority Over-sampling Technique) is a powerful oversampling technique that can be used to address class imbalance in machine learning datasets. When combined with a deep learning algorithm such as CNN (Convolutional Neural Network), SMOTE can help improve the performance of the classifier on the minority class by creating synthetic samples that better represent the underlying distribution of the data.

4. Training and Testing Subset: As the dataset is imbalanced, many classifiers show bias for majority classes. The features of minority class are treated as noise and are ignored. Hence it is proposed to select a sample dataset.

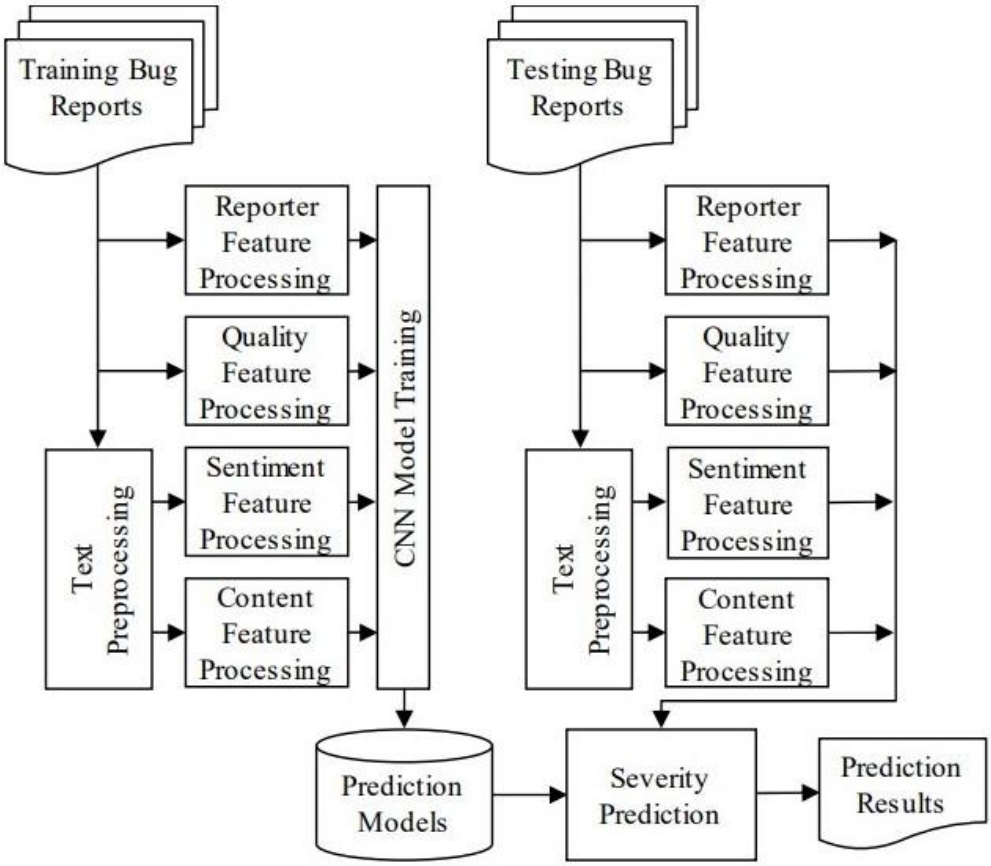

Fig 2 Training and Testing subset

5. Applying algorithm: Here, we have used ***Convolution Neural Network (CNN).*** Convolution Neural Networks are a powerful tool for bug prediction, as they can identify patterns and trends in that code that are difficult for humans to detect. One of the advantages of using CNN is their ability to handle large and complex datasets, and also their ability to adopt and learn overtime. CNN can update their models to reflect the latest trends and patterns, this allows CNNs to remain more accurate and effective overtime even if new types of bugs and issues emerge.

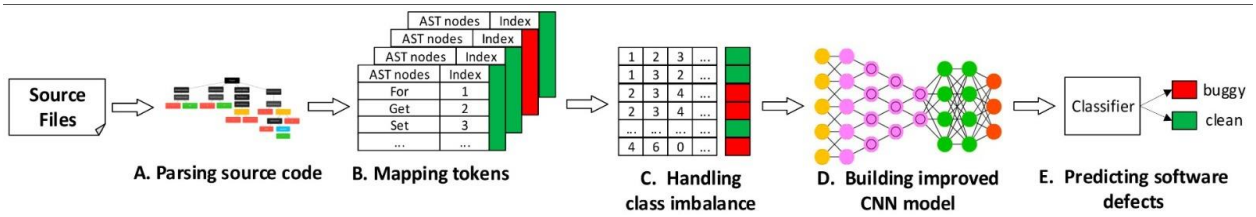

Fig 3: Applying Algorithm

6. Predicting results: When comparing CNNs and SVMs for bug prediction, CNNs have several advantages over SVMs. Firstly, CNNs can handle large and complex datasets, allowing them to identify patterns and trends in the data that may be difficult to detect using traditional SVM-based methods. Additionally, CNNs are highly customizable, allowing developers to adjust the model parameters to optimize for specific use cases and data..The test subset is applied on the trained model.The metrices used is accuracy. The graph is plotted and the desirable results are achieved.

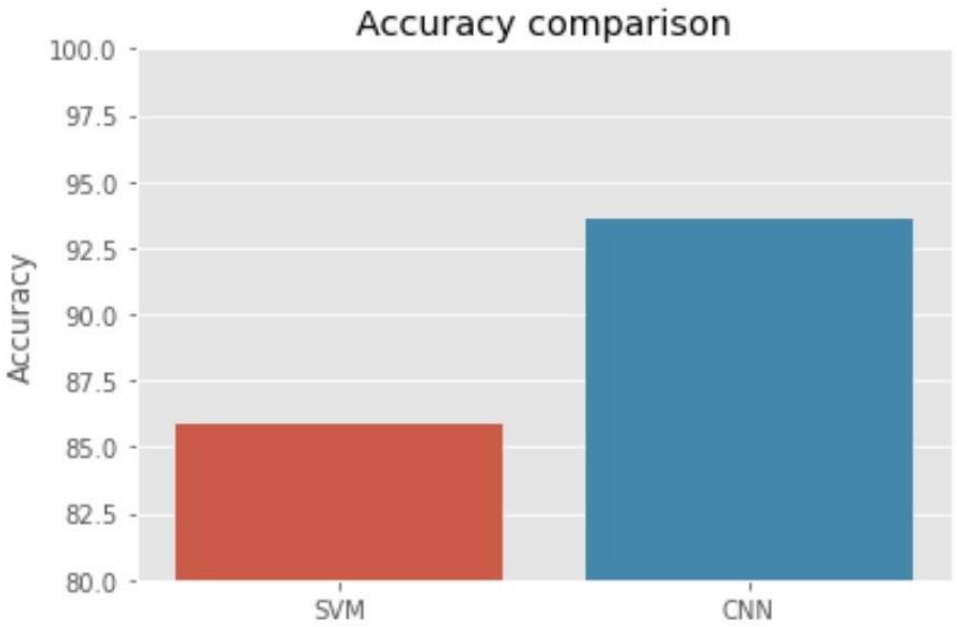

Fig 4: Predicting Results

VI.Conclusion

In this project we have reviewed on the technologies which are being used for finding and improving bug tracking system. Further we have introduced different techniques used to implement them. Present methods include database server and admin information. Later on we use SVM and CNN are used for comparative study in terms of accuracy and storing of structure data into the database etc. This comparison will help us in building our system more convenient and useful. From the research we have proposed the system which will predict time required for particular task.

VII.Future Scope

The proposed work can be extended using advanced text pre-processing techniques for optimizing the clustering and classification work, and also modern text clustering and classification algorithms can be implemented and compared with the proposed algorithm. We also plan to extend it by performing feature selection on our factors. Employing feature selection may improve the performance of our models since it removes redundant factors. Our results show that blocking bugs take longer to be fixed than non-blocking bugs; however it is unclear if blocking bugs require more effort and resources than non-blocking bugs. To tackle this question, we plan to link bug reports with information from the version control systems, leverage metrics at commit level and perform a quantitative analysis that may help us to confirm or refute our intuition that blocking bugs indeed require more effort.

REFERENCES:

[1] Current challenges in automatic software repair . Goues, Claire Le, Forrest, Stephanie and Weimer, Westley. New York: Springer Science+Business Media, 2013.
[2] Yuan Tian, David Lo, Chengnian Sun, "Information Retrieval Based Nearest Neighbour Classification for FineGrained Bug Severity Prediction", WCRE, 2012, 2013 20th Working Conference on Reverse Engineering (WCRE), 2013 20th Working Conference on Reverse Engineering (WCRE) 2012, pp. 215-224, doi:10.1109/WCRE.2012.31
[3] http://www.bugzilla.org/
[4] Shaffiei, Zatul Amilah, Mudiana Mokhsin, and Saidatul Rahah Hamidi. "Change and Bug Tracking System: Anjung Penchala Sdn. Bhd." Change 10.3 (2010).
[5] http://www.techopedia.com/definition/13698/tokenization
[6] Valdivia Garcia, Harold, and Emad Shihab. "Characterizing and predicting blocking bugs in open source projects." Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.
[7] N. Kumar Nagwani and S. Verma, "CLUBAS: An Algorithm and Java Based Tool for Software Bug Classification Using Bug Attributes Similarities," Journal of Software Engineering and Applications, Vol. 5 No. 6, 2012, pp. 436-447. doi: 10.4236/jsea.2012.56050.
[8] http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/
[9]http://users.ece.cmu.edu/~koopman/des_s99/sw_testing/
[10] http://istqbexamcertification.com/what-is-the-cost-of-defects-in-software-testing/
[11] Wang, Hui. "Software Defects Classification Prediction Based On Mining Software Repository." (2014).
[12] http://istqbexamcertification.com/what-is-the-difference-between-severity-and-priority/