SOFTWARE BUG PREDICTION USING MACHINE LEARNING

[1]Dr. K.L.S. Soujanya ,[2]S. Balaji Manikanta Sai ,[3]V. Koushik Yadav ,[4]D. Navaneeth Sai & [5]M. Avinash

[1]Professor, Department of Information Technology, CMR College of Engineering & Technology

[2, 3, 4, 5] B-Tech, Department of Information Technology, CMR College of Engineering &

Abstract:

Software Bug Prediction (SBP) is an important process in software development and maintenance, which concerns the overall software success. This is because predicting the software faults in the earlier phase improves the software quality, reliability, and efficiency and reduces the software cost. However, developing a robust bug prediction model is challenging, and many techniques have been proposed in the literature. This paper presents a software bug prediction model based on machine learning (ML) algorithms. Three supervised ML algorithms have been used to predict future software faults based on historical data. These classifiers are Naïve Bayes (NB), Decision Tree (DT), and Artificial Neural Networks (ANNs). The evaluation process showed that ML algorithms can be used effectively with a high accuracy rate. Furthermore, a comparison measure is applied to compare the proposed prediction model with other approaches. The collected results showed that the ML approach has a better performance.

INTRODUCTION:

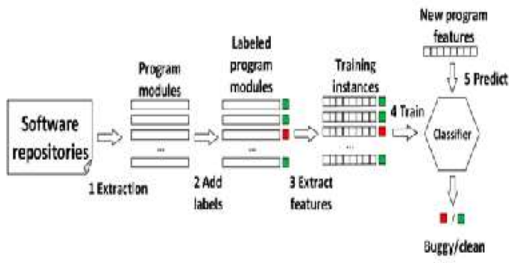
Problem Statement: Software Bug Prediction (SBP) is a critical process in software development and maintenance, that is concerned with the overall success of software. This is because predicting software faults earlier in the development process improves software quality, reliability, and efficiency reducing software costs. However, creating a robust bug prediction model is a difficult task, and numerous techniques have been proposed in the literature. This paper presents a machine learning (ML)-based software bug prediction model. Based on historical data, three supervised ML algorithms were used to predict future software faults. These classifiers include Nave Bayes (NB), Decision Trees (DT), and Convolutional Neural Networks (CNN). The evaluation process demonstrated that ML algorithms can be used effectively and accurately.

OBJECTIVES:

• To find the Accuracy of Bug Detection by using Machine Learning Algorithms.

• Prediction of Bugs occurring before the development of the Project.

IMPLEMENTATION:



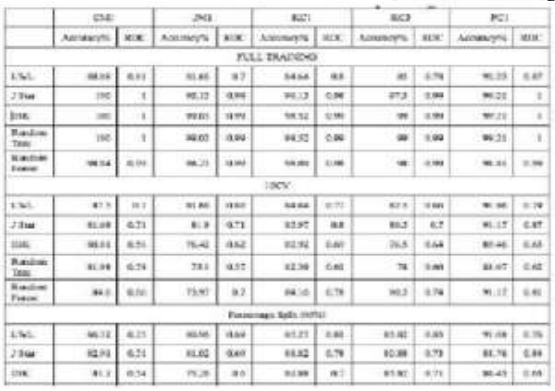
• Several works propose feature identification techniques. However, few works attempt to compare features with one another.

• In their precursor work, Wilde and Scully propose a technique to identify features by analysing execution traces of test cases.

• They use two sets of test cases to build two execution traces: An execution trace where functionality is exercised; An execution trace where the functionality is not.

• Then, they compare execution traces to identify the feature associated with the functionality in the program.

• In their work, the authors only use dynamic data to identify features, no static analysis of the program is performed. In their work, the authors only use dynamic data to identify features, no static analysis of the program is performed.



	CM1		JM1		KC1		KC3		PC1	
	Accuracy%	ROC	Accuracy%	ROC	Accuracy%	ROC	Accuracy%	ROC	Accuracy%	ROC
FULL TRAINING										
LWL	88.95	0.91	91.46	0.7	84.64	0.8	85	0.78	91.23	0.87
J 48	100	1	98.12	0.99	99.13	0.98	97.3	0.99	99.21	1
IBK	100	1	99.81	0.99	98.52	0.99	99	0.99	99.21	1
Random Tree	100	1	98.67	0.99	98.52	0.98	99	0.99	99.21	1
Random Forest	99.54	0.99	98.21	0.99	99.00	0.98	98	0.99	98.44	0.99
10CV										
LWL	87.5	0.7	91.86	0.69	84.64	0.77	82.5	0.61	91.98	0.78
J 48	81.08	0.71	81.9	0.71	82.97	0.8	80.5	0.7	91.17	0.87
IBK	80.81	0.51	76.42	0.62	82.92	0.60	76.5	0.64	89.46	0.65
Random Tree	81.08	0.74	73.1	0.57	82.38	0.61	78	0.69	83.07	0.62
Random Forest	84.0	0.81	73.97	0.7	84.16	0.78	80.5	0.76	91.17	0.81
Percentage Split (66%)										
LWL	86.72	0.73	80.56	0.69	85.23	0.80	83.82	0.83	91.08	0.76
J 48	82.91	0.71	81.02	0.69	83.82	0.79	80.88	0.73	81.76	0.88
IBK	81.3	0.54	75.28	0.6	84.88	0.7	83.82	0.71	88.43	0.66

Table 1: Feature Identification

• This approach decomposes into a process and a set of tools to support the process.

• The process makes use of processor emulation, knowledge filtering, probabilistic ranking, and model transformations to support the analysis of large multi-threaded object-oriented programs and to deal with imprecision and noise.

• They use data collected from the realization of functionality, under different scenarios, to filter static data modeled as class diagrams, thus relating classes with features and with scenarios, and highlighting differences among features.

• Maintainers can use this approach to build and compare micro-architectures to precisely locate responsibilities and feature differences.

PROPOSED SYSTEM :

Objective of Proposed Model:

The proposed system is a software bug prediction model based on machine learning algorithms. It uses historical data to predict future software faults to improve software quality, reliability, and efficiency, and reduce software costs. The system uses three supervised machine learning classifiers: Naïve Bayes, Decision Tree, Random Forest, and Convolutional Neural Networks. The evaluation process of the system showed high accuracy rates and effectiveness in predicting software faults. Additionally, a comparison measure was applied to compare the proposed prediction model with other approaches, and the results showed that the machine learning approach outperformed other methods. Overall, the proposed system is a robust and effective way to predict software bugs and improve the software development and maintenance process. It can be used by software developers and maintenance teams to enhance the quality, reliability, and efficiency of software systems while reducing costs.

The proposed system of software bug prediction based on machine learning algorithms has several advantages, including:

➢ Improved software quality: By predicting software bugs in advance, the proposed system can help software developers to identify and fix the issues before the software is deployed. This can lead to a higher quality software product that meets user requirements and expectations.

➢ Cost savings: Identifying and fixing software bugs can be time-consuming and expensive. By predicting bugs and addressing them early in the development process, the proposed system can help reduce the overall cost of software development and maintenance.

➢ Faster development: With the help of the proposed system, software developers can identify and fix bugs earlier in the development cycle, which can help accelerate the development process and reduce time to market.

➢ Scalability: The proposed system is based on machine learning algorithms, which can learn and improve over time as more data becomes available. This means that the system can be scaled up to handle larger and more complex software projects. Overall, the proposed system offers a range of advantages that can help software development teams to improve their software quality, reliability, and efficiency while reducing costs and time to market.

RESULTS:

Comparison of Existing Solutions: There are many studies about software bug prediction using machine learning techniques. For example, the study in [2] proposed a linear Auto-Regression (AR) approach to predict faulty modules. The study predicts the software's future faults depending on the historical data of the software's accumulated faults. The study also evaluated and compared the AR model with the Known power model (POWM) using Root Mean Square Error (RMSE) measure. In addition, the study used three datasets for evaluation and the results were promising. The studies in [1], and [2] analyzed the applicability of various ML methods for fault prediction. The most important previous research about each ML technique and the current trends in software bug prediction using machine learning. This study can be used as ground or step to prepare for future work in software bug prediction. Malhotra Ruchika in [9] presented a good systematic review of software bug prediction techniques, using Machine Learning (ML). The paper included a review of all the studies between the period 1991 and 2013, analyzed the ML techniques for software bug prediction models, assessed their performance, compared ML and statistic techniques, compared different ML techniques, and summarized the strength and weaknesses of the ML techniques. In [10], the paper provided a benchmark to allow for common and useful comparisons between different bug prediction approaches. The study presented a comprehensive comparison between a well-known bug prediction approach, also introduced a new approach, and evaluated its performance by building a good comparison with other approaches using the presented benchmark. The study in [9] assessed various object-oriented metrics by used machine learning techniques (decision tree and neural networks) and statistical techniques (logical and linear regression). The results of the study showed that the Coupling Between Object (CBO) metric is the best metric to predict the bugs in the class and the Line of Code (LOC) is fairly well, but the Depth of Inheritance Tree (DIT) and Number of Children (NOC) are untrusted metrics. Singh and Chug [9] discussed five popular ML algorithms used for software defect prediction i.e, Artificial Neural Networks (ANNs), Particle Swarm Optimization (PSO), Decision Tree (DT), Naïve Bayes (NB) and Linear Classifiers (LC). The study presented important results including that the ANN has lowest error rate followed by DT, but the linear classifier is better than other algorithms in term of defect prediction accuracy, the most popular methods used in software defect prediction are: DT, BL, ANN, SVM, RBL and EA, and the common metrics used in software defect prediction studies are: Line

Of Code (LOC) metrics, object oriented metrics such as cohesion, coupling and inheritance, also other metrics called hybrid metrics which used both object oriented and procedural metrics, furthermore the results showed that most software defect prediction studied used NASA dataset and PROMISE dataset. Moreover, the studies in [08], [9] discussed various ML techniques and provided the ML capabilities in software defect prediction. The studies assisted the developer to use useful software metrics and suitable data mining technique in order to enhance the software quality. The study in determined the most effective metrics which are useful in defect prediction such as Response for class (ROC), Line of code (LOC) and Lack of Coding Quality (LOCQ). Bavisi et al. presented the most popular data mining technique (k-Nearest Neighbors, Naïve Bayes, C4.5 and Decision trees). The study analyzed and compared four algorithms and discussed the advantages and disadvantages of each algorithm. The results of the study showed that there were different factors affecting the accuracy of each technique; such as the nature of the problem, the used dataset and its performance matrix

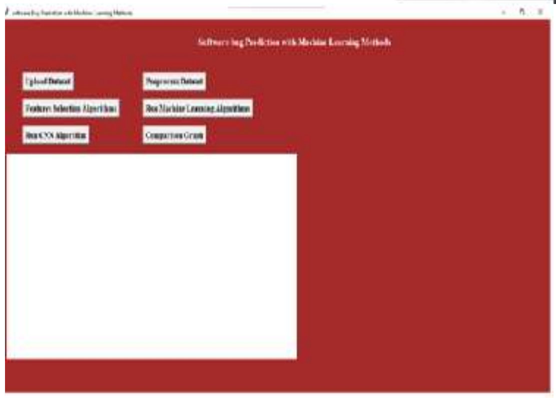

Software Bug Prediction with Machine Learning Methods
Upload Dataset
Preprocess Dataset
Features Selection Algorithms
Run Machine Learning Algorithms
Run CNN Algorithm
Comparison Graph

Fig: 2. UI of our Project

• The UI of the project is designed in such a way that it's user-friendly and all the functionalities are labeled to perform various operations over a Software project.

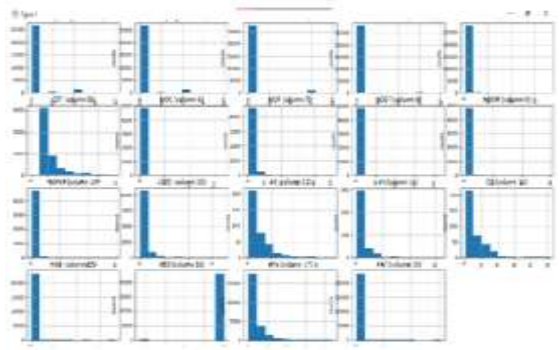


Fig: 3. Evaluation of Dataset based on different attributes

• For the Evaluation process of the dataset we need to consider few attributes that which are provided in the Dataset and it will show us the evaluation of those selected attributes in a dataset in the form pf Bar-Graphs.

CONCLUSION

To locate a bug, developers use not only the content of the bug report but also domain knowledge relevant to the software project. We introduced a learning-to-rank approach that emulates the bug-finding process employed by developers. The ranking model characterizes useful relationships between a bug report and source code files by leveraging domain knowledge, such as API specifications, the syntactic structure of code, or issue tracking data. Experimental evaluations on six Java projects show that our approach can locate the relevant files within the top 10 recommendations for over 70 percent of the bug reports in Eclipse Platform and Tomcat. Furthermore, the proposed ranking model outperforms three recent state-of-the-art approaches. Feature evaluation experiments employing greedy backward feature elimination demonstrate that all features are useful. When coupled with runtime analysis, the feature evaluation results can be utilized to select a subset of features to achieve a target trade-off between system accuracy and runtime complexity.

FUTURE ENHANCEMENT

In future work, we will leverage additional types of domain knowledge, such as the stack traces submitted with bug reports and the file change history, as well as features previously used in defect prediction systems. We also plan to use the ranking SVM with nonlinear kernels and further evaluate the approach on projects in other programming languages

REFERENCES:

[1] G. Antoniol and Y.-G. Gueheneuc, “Feature identification: A novel approach and a case study,” in Proc. 21st IEEE Int. Conf. Softw. Maintenance, Washington, DC, USA, 2005, pp. 357–366.

[2] A. Sheta and D. Rine, “Modeling Incremental Faults of Software Testing Process Using AR Models”, the Proceeding of 4th International Multi Conferences on Computer Science and Information Technology (CSIT 2006), Amman, Jordan. Vol. 3. 2006.0

[3] A. Bacchelli and C. Bird, “Expectations, outcomes, and challenges of modern code review,” in Proc. Int. Conf. Softw. Eng., Piscataway, NJ, USA, 2013, pp. 712–721.

[4] J. Zhou, H. Zhang, and D. Lo, “Where should the bugs be fixed? - more accurate information retrieval-based bug localization based on bug reports,” in Proc. Int. Conf. Softw. Eng., Piscataway, NJ, USA, 2012 pp. 14–24.

[5] R. M. Bell, T. J. Ostrand, and E. J. Weyuker, “Looking for bugs in all the right places,” in Proc. Int. Symp. Softw. Testing Anal., New York, NY, USA, 2006, pp. 61–72.

[6] D. Sharma and P. Chandra, "Software Fault Prediction Using Machine Learning Techniques," Smart Computing and

Informatics. Springer, Singapore, 2018. 541-549.

[7] T. J. Biggerstaff, B. G. Mitbander, and D. Webster, “The concept assignment problem in program understanding,” in Proc. 15th Int. Conf. Softw. Eng., Los Alamitos, CA, USA, 1993, pp. 482–498.

[8] D. Binkley and D. Lawrie, “Learning to rank improves IR in SE,” in Proc. IEEE Int. Conf. Softw. Maintenance Evol., Washington, DC, USA, 2014, pp. 441–445.

[9] Malhotra, Ruchika. "A systematic review of machine learning techniques for software fault prediction." Applied Soft Computing 27 (2015): 504-518.

[10] D'Ambros, Marco, Michele Lanza, and Romain Robbes. "An extensive comparison of bug prediction approaches." Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on. IEEE, 2010.

[11] Reddy, b. V. R., dasari, n., & venkateswararao, k. (2021). A steganography system with gausian markov random fields and error detection codes.

[12] Soujanya, K. (2018). Ontology based variability management for dynamic reconfiguration of software product lines. Journal of Advanced Research in Dynamical and Control Systems, 9(18), 2361-2375.

[13] Soujanya, K. L. S., & AnandaRao, A. (2016). A generic framework for configuration management of spl and controlling evolution of complex software products. ACM SIGSOFT Software Engineering Notes, 41(1), 1-10.

[14] Kommanaboyina, L. S., & Akepogu, A. R. (2016). Change Propagation in Software Product Lines Using ACP Algorithm. In Transactions on Engineering Technologies: International MultiConference of Engineers and Computer Scientists 2015 (pp. 95-108). Springer Singapore.

[15] Vinay, R., K. L. S. Soujanya, and P. Singh. "Disease prediction by using deep learning based on patient treatment history." Int. J. Recent Technol. Eng 7.6 (2019): 745-754.

[16] Latha, C. M., Bhuvaneswari, S., & Soujanya, K. L. S. (2022, November). Stock Price Prediction using HFTSF Algorithm. In 2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 1053-1059). IEEE.