iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VI **Month of publication:** June 2023

DOI: https://doi.org/10.22214/ijraset.2023.54487

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com

Detecting Bugs in Software using Supervised Machine Learning Approaches

V. Bharathi[1], K.Bala Krishna [2], N.Srinivas [3]
[1]*Associate Professor,* [2, 3]*Assistant Professor, Department of CSE, Geethanjali Institute of Science & Technology, Nellore, A.P.*

Abstract: Software Flaw Projection (SFP) is an important issue in software development and maintenance process. Software flaws can cause significant problems for software development teams. So, projecting the software faults in earlier phase improves the software quality, reliability, efficiency and reduces the software cost. However, developing robust flaw projection model is a challenging task and many techniques have been proposed. Projecting the likelihood of flaws occurring in software can help developers prevent or mitigate their impact. This paper presents a software flaw projection model based on Machine Learning (ML) algorithms. Supervised ML algorithms have been used to predict future software faults based on historical data. The evaluation process proved that ML algorithms can be used effectively with 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.
Keywords: Software Quality, Naïve Bayes, Fault, XG Boost

I. INTRODUCTION

In today's fast paced software development world, software flaws are a common occurrence. These flaws can cause significant problems such as crashes, data corruption and security breaches. Detecting and fixing software flaws is a crucial task in the software development process. Traditionally, software developers and testers rely on manual testing and debugging to detect and fix software flaws. However, this process can be time consuming and error-prone. With the recent advancements in machine learning, it is possible to use Machine Learning algorithms to project software flaws automatically. Supervised Machine Learning approach for software flaw projection is a process of analyzing software metrics and features to predict the likelihood of software flaws. This approach has the potential to improve software quality by detecting flaws before they cause significant problems. Machine Learning has emerged as a powerful tool in predicting software flaws, providing developers with the ability to identify and fix potential issues before they occur.

Supervised Machine Learning approach for software flaw projection is a technique used to identify potential flaws in software before they occur. With the increasing complexity of software applications, detecting bugs manually is becoming increasingly difficult and time-consuming. Machine learning algorithms can analyze historical data from past software projects to predict potential bugs in new code. The process involves gathering data related to the software development process, such as code changes, code complexity, and the time taken to complete a task. This data is then fed into a machine learning model, which learns from the patterns and relationships in the data to predict where flaws are likely to occur. The benefits of using machine learning for flaw projection include faster detection of potential issues, improved software quality, and reduced costs associated with fixing flaws. This technique can be applied to various stages of the software development process, including requirements gathering, design, coding, and testing.

The process typically involves the following steps:

1) *Data Collection:* Collecting historical data on past software flaws, information about the software application, the code, and the circumstances surrounding the flaw.
2) *Feature Extraction:* Extracting features from the data that are relevant to predicting flaws. These might include metrics like code complexity, code churn (i.e., how frequently the code is changed), and the number of developers involved in writing the code.
3) *Model Training:* Using machine learning algorithms to train models on the extracted features. The models are trained to recognize patterns and relationships between the features and past bugs.
4) *Model Evaluation:* Evaluating the performance of the trained models on new data to see how accurately they can predict bugs.
5) *Flaw Projection:* Using the trained models to project potential flaws in new software code before they occur.

However, supervised Machine Learning approach for software flaw projection can help software developers catch and fix flaws early in the development process, reducing the overall cost and effort required to maintain and improve software applications.

II. LITERATURE SURVEY

In the literature, there are also more sophisticated approaches focusing on predicting software bugs. In recent years, various Machine Learning techniques have been applied to software flaw projection, including decision trees, random forests, support vector machines, and neural networks. These techniques can analyze code metrics, change history, and other software characteristics to predict the likelihood of bugs occurring in a particular piece of code. By using ML models, software developers can identify potential issues early in the development process and take corrective action.

1) Software Bug Prediction Using Machine Learning Approach

Software Bug Prediction (SBP) is an important issue in software development and maintenance processes, which concerns with the overall of software successes. This is because predicting the software faults in earlier phase improves the software quality, reliability, efficiency and reduces the software cost. However, developing robust bug prediction model is a challenging task and many techniques have been proposed in the literature. Concurrently, they proposed a software bug prediction model based on Machine Learning Algorithms. The supervised Machine Learning algorithms have been used to predict the future software faults based on historical data. The evaluation process shows that ML algorithms can be used effectively with high accuracy rate.

2) A Developer Centered Bug Prediction Model

Bug prediction techniques are used to identify areas of software systems that are more likely to contain bugs. These prediction models represent an important aid when the resources available for testing are scarce, since they can indicate where to invest such resources. The Scientific community has developed several bug prediction models that can be classified into two families, based on the information they exploit to discriminate between buggy and clean code components. The first set of techniques exploits product metrics while the second one focuses on process metrics. These two measures consider the amount of code components a developer modifies in a given time period and predict the bugs efficiently.

3) Prediction of software defect using supervised Machine Learning practices

Defects are common in software systems and can potentially cause various problems to software users. Different methods have been developed to quickly predict the most likely locations of defects in large code bases. Most of them focus on designing features (e.g. complexity metrics) that correlate with potentially defective code. Those approaches however do not sufficiently capture the syntax and different levels of semantics of source code, an important capability for building accurate prediction models. In our approach, three supervised Machine Learning algorithms are considered to build the model and predict the occurrence of the soft++ware bugs based on historical data by deploying the classifiers Logistic regression, Naïve Bayes, and Decision Tree. Historical data has been used to predict the future software faults by deploying the classifier algorithms and make the models a better choice for predictions using random forest ensemble classifiers and validating the models with K-Fold cross validation technique which results in the model effectively working for all the scenarios.

III. METHODOLOGY

The algorithms used are :
LR – Logistic regression
NB – Naive bayes
DT – Decision tree
RF – Random forest
XGBoost

A. Logistic Regression

Logistic Regression is a type of statistical model which is often used for classification and predictive analysis. It is also called as binary classifier. Logistic Regression estimates the probability of an event occurring. It is an example of supervised learning. It is used to calculate or predict the probability of a binary event occurring.

B. Naive Bayes Algorithm

Naive bayes algorithm is a supervised learning algorithm, which is based on bayes theorem and used for solving classification problems. Naive Bayes classifier is one of the simple and most effective classification algorithms which helps building the fast Machine Learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. The algorithm uses a probabilistic approach to predict the class label of a given input based on the probability distribution of the features. The key advantages of the algorithm is that it requires a relatively small amount of training data to make accurate predictions.

C. Decision tree algorithm

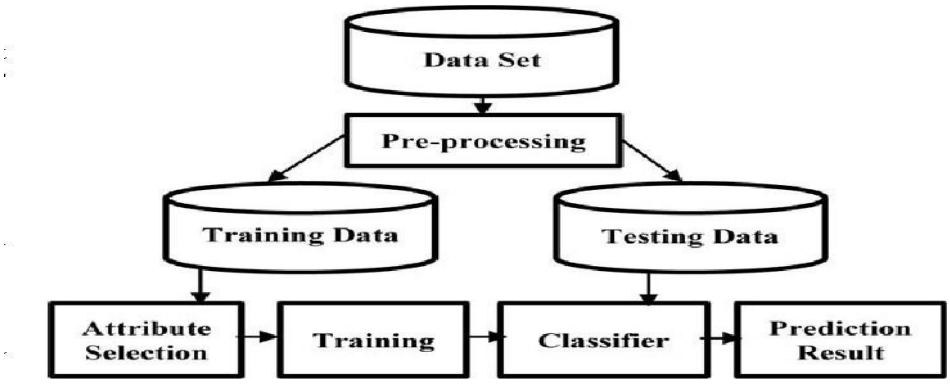
Decision tree is a supervised learning technique that can be used for both classification and regression problem but mostly is preferred for solving classification problems. It is a tree structured classifier where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes which are the decision node and the leaf node. Decision nodes are used to make any decision and have multiple branches, whereas leaf nodes are the output of those decisions and do not contain any further branches.

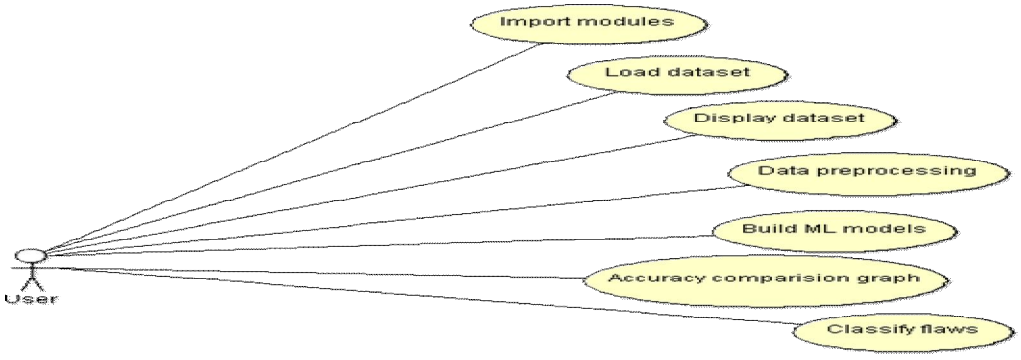
D. Random Forest Algorithm

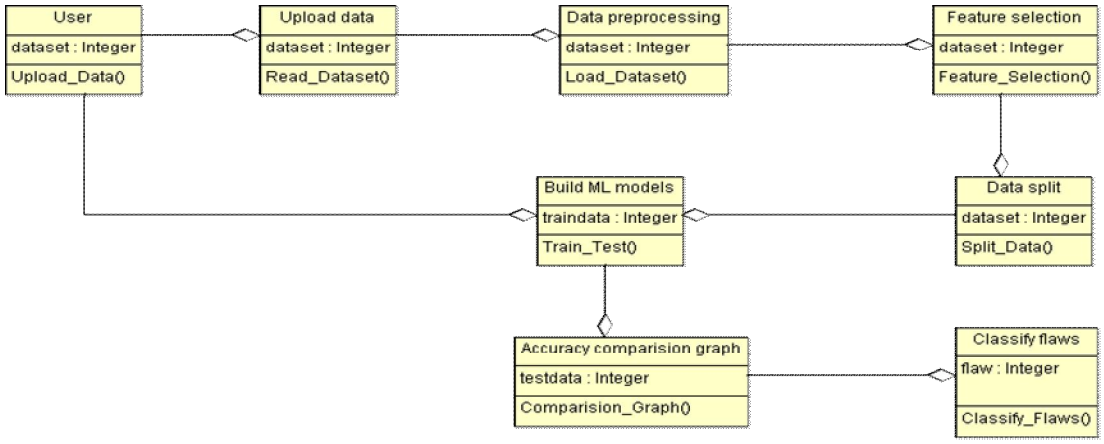
Random forest is a bagging technique and not a boosting technique. The trees in random forests are run in parallel. There is no interaction between these trees while building the trees. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes(classification) or mean prediction(regression) of the individual trees. A random forest is a meta-estimator(i.e. it combines the results of multiple predictions).It is a combination of multiple decision trees which combines the output of multiple decision trees to reach a single result.

E. XGBoost Algorithm

An XGBoost(Extreme Gradient Boosting) is a machine learning algorithm used for classification and regression problems. The key idea behind XG Boost is to use an ensemble of decision trees where each tree is built to correct the mistakes of the previous tree. The algorithm trains a sequence of weak models(decision trees) and iteratively improves them by minimizing a loss function.







IV. RESULTS AND SCREENSHOTS

The steps followed in the execution process are shown in the attached screenshot pictures:

1) Open the command prompt (figure6.6.1).
2) Enter the command to view the display to perform algorithms (figure 6.6.2).
3) Open the dataset(figure 6.6.3).
4) In the display click on load dataset to load the dataset(figure 6.6.4).
5) Once the data gets loaded click on pre-process data for performing data pre- processing
6) Click on the Decision tree to get the accuracy(figure 6.6.6).
7) Click on the Random forest to get the accuracy(figure 6.6.7).
8) Click on the Logistic regression to get the accuracy(figure 6.6.8).
9) Click on the Naïve bayes to get the accuracy(figure 6.6.9).
10) Click on the XGBoost to get the accuracy(figure 6.6.10).
11) Click on Accuracy comparison to get the graph that compares the accuracies of all the algorithms

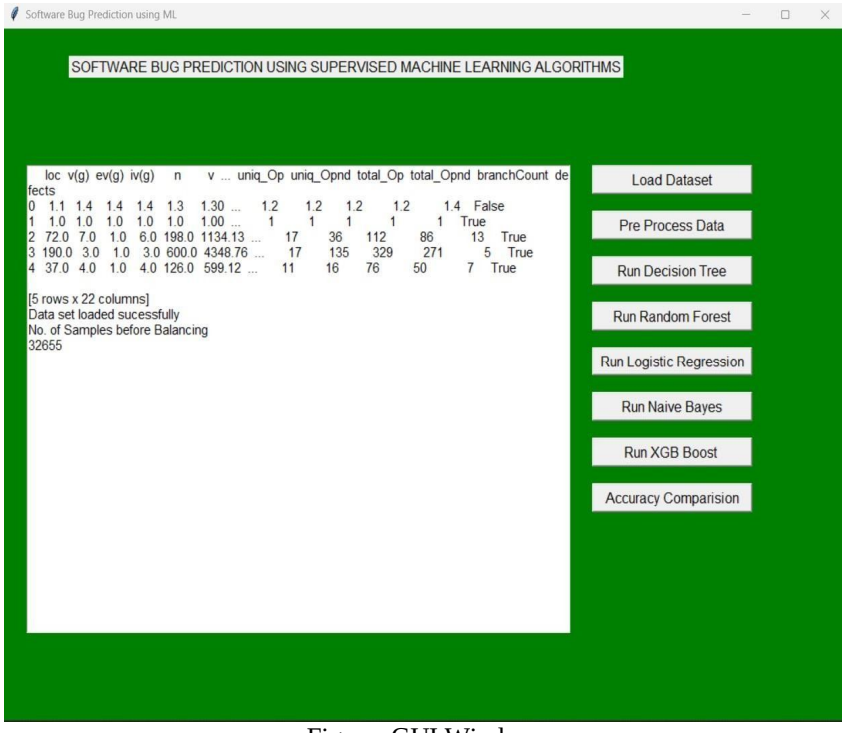


Figure: GUI Window

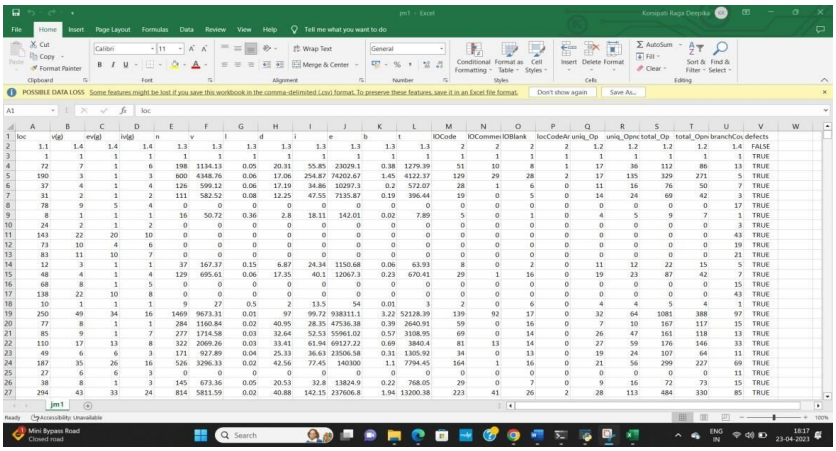


Figure: Dataset Details

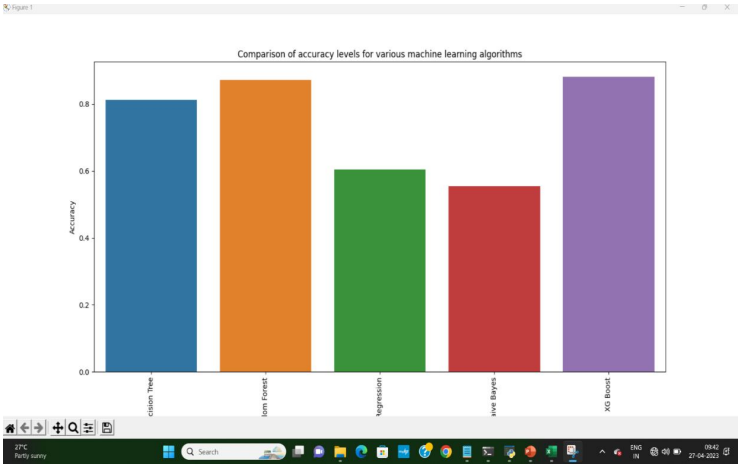


Figure: Comparison Results of all models

V. CONCLUSIONS

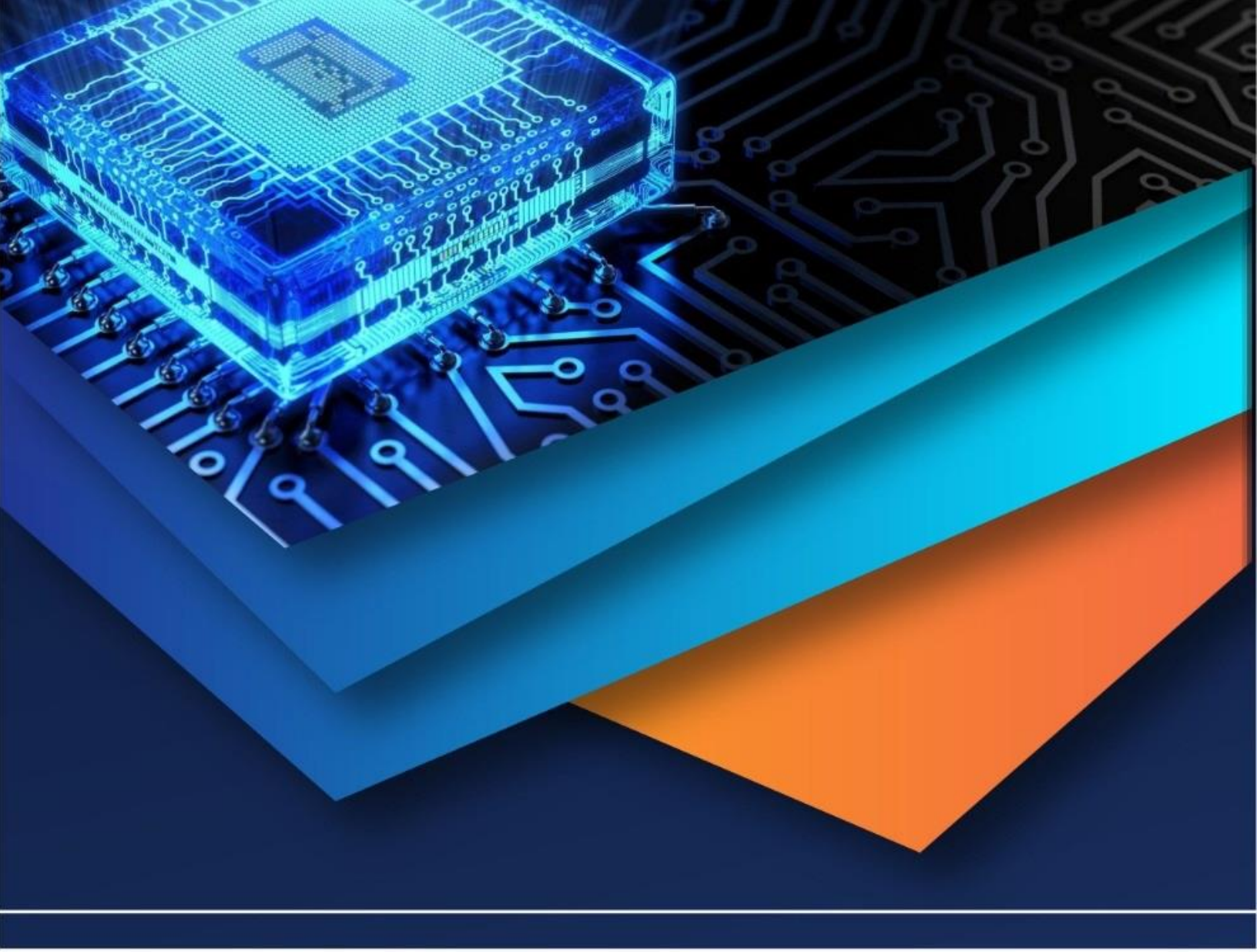
Supervised Machine Learning approach for software flaw projection is a promising approach that has gained attention in recent years. Overall, the effectiveness of software flaw projection depends on the quality of the data used to train the models and the features selected to represent the software. Additionally, the choice of machine learning algorithms can have an impact on the accuracy and reliability of the predictions. Despite some limitations and challenges, supervised machine learning approach for software flaw projection has the potential to significantly improve the quality of software and reduce the cost of development and maintenance. The evaluation process is implemented using the dataset .Experimental results are collected based on accuracy , precision , recall ,F-measure .The results revealed that the used algorithms are efficient to predict the software flaws. The comparison results showed that XG Boost has the best result over the others. Moreover, ML approach provides a better performance for the projection of flaws.

After the comparative analysis of the various Supervised Machine Learning models, we can infer that the XG Boost Model is the best approach to be used for projecting software flaws. Among all the supervised machine learning algorithms used, XG Boost has highest accuracy. Hence, we conclude that the XG Boost is an efficient model among all the algorithms used. But we have not achieved 100 % efficiency hence to improve the performance of the model we can use other ML techniques like ANN and provide an extensive comparison among them.

REFERENCES

[1] Dario Di Nucci, Fabio Palomba ,Giuseppe De Rosa Gabriele Bavota ,Rocco Oliveto, and Andrea De Lucia, “A developer centric bug prediction model", IEEE Transactions on Software Engineering, Vo.l 44, Issue 1, pp. 5-24, 2018.

[2] F. Wu et al., "Cross-Project and Within-Project Semi supervised Software Defect Prediction: A Unified approach", IEEE Transactions on Reliability, pp. 1-17, 2018.
[3] Meiliana, S. Karim, H. L. H. S. Warnars, F. L. Gaol, E. Abdurachman and B. Soewito, "Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset", In Proc. IEEE International Conference on Cybernetics and Computational Intelligence(CyberneticsCom), Phuket, pp.19-23, 2017.
[4] M. M. Rosli, N. H. I. Teo, N. S. M. Yusop, and N. S. Mohammad, "The design of a software fault prone application using evolutionary algorithm," in Proc. IEEE Conference on Open Systems (ICOS 2011). Los Alamitos, California: IEEE Computer Society, pp. 38-343. 2011.
[5] D'Ambros, M. Lanza, and R. Robbes, "An Extensive Comparison of Bug Prediction Approaches", In Proc. IEEE Seventh Working Conf. Mining Software Repositories, pp. 31-41, 2010
[6] Pushphavathi T P, "An Approach for Software Defect Prediction by Combined SoftComputing", In Proc, International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS) pp.3003-3006, 2017.
[7] Kumar, Lov, and Ashish Sureka. "Aging Related Bug Prediction using Extreme Learning Machines.", In Proc. 14th IEEE India Council International Conference (INDICON), pp.1-6, IEEE, 2017.
[8] Nigam, Ayan, et al. "Classifying the bugs using multi-class semi supervised support vector machine.", In Proc. International Conference, Pattern Recognition, Informatics and Medical Engineering (PRIME), pp.393-397, IEEE, 2012.
[9] Gyimothy, T., Ferenc, R. and Siket, I., "Empirical validation of object-oriented metrics on open source software for fault prediction", IEEE Transactions on Software Engineering, 31(10), pp. 897-910, 2005.
[10] John T. Pohlmann and Dennis w. Leitnera "Comparison of Ordinary Least Squares and Logistic Regression", The Ohio Journal of Science. vol. 103, number 5, pp. 118- 125, Dec, 2003.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY