International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-9 Issue-2, July 2020

Analysis on Detecting a Bug in a Software using Machine Learning

Rashmi P, Prashanth Kambli

Abstract: ***In today's scenario, it is very essential in the development phase of a software, predicting a bug and to obtain a successful software. This can be achieved only through predicting some of the faults in the earlier phase itself such that, it can lead to have a reliable, efficient and a quality software. The challenging task here is to have a well sophisticated model that can predict the bug leading to a cost-effective software. In order to achieve this, few machine learning algorithms are used that produce accuracy with trained and test datasets. A variety of machine learning methodologies have been developed to learn and detect a bug in a software. In this paper, we perform the analysis on detecting a bug in a software using machine learning methods which is very much useful for Software Industries. It summarizes the existing work on detecting a bug in a software by providing the information about various methods involved in bug prediction and points out at the accuracy obtained by the existing methods, advantages, and the drawbacks while working with bug prediction.***

Keywords: Software Development, Software Bug Prediction, Quality Software, Machine Learning, Accuracy.

I. INTRODUCTION

Machine learning is the ability to learn and predict an event based on experience. The machine learning algorithms is mainly classified into supervised and unsupervised algorithms. Supervised algorithms will consist of an output variable and we search for similarity between dependent and independent variables. Unsupervised algorithm is where we transfer the data and predict the result. The algorithms utilize the whole data and produce the results. Machine learning algorithm is also used to predict uncertainty [1]. Some of the Machine Learning algorithms are used in order to predict a bug in software mainly during the development phase of life cycle mainly identification of problem, planning phase, design, development, testing, deploy and maintenance and models of software development life cycle. With the help of Machine learning algorithms and statistical analysis, bug prediction can be performed. Different methodologies are used to increase the quality during software development process [2]. Whenever a system fails there exists a software bug. Software bug is caused due to developers and team members. In order to improvise the quality of software, to reduce the budget and time saving, software bug prediction is required.

Manuscript received on May 25, 2020.
Revised Manuscript received on June 29, 2020.
Manuscript published on July 30, 2020.
***** Correspondence Author
Rashmi P, Department of CSE, MSRIT, Bangalore, India.
Prashanth Kambli Assistant professor, Department of MSRIT, IS&E, Bangalore, India.

It finds the module that have bugs and also performs testing. Identification of the bug at the early stage will lead to effective allocation of resources, reduces the time and cost of developing a software and thus results in a high-quality software. Therefore, Software Bug Prediction model is necessary in understanding, evaluating and improvising the quality of a software system [3].

The author has discussed about few commonly used Machine learning algorithms. Considering the applications, discussion is made regarding the pros and cons of the machine learning algorithms, that's required to make right decision for selecting appropriate algorithm based on requirement specification. [4]. The software design complexity is increasing due to software defects. Testers improve the quality of software by fixing the bugs, so the analysis of bugs will improve the efficiency. As the number of bugs reduces, analysis that is performed manually cannot meet the needs of defect analysis. To solve this automatic defect classification method was proposed. The defect analysis will increase the efficiency and reduce software maintenance cost [5].Software bug prediction is performed using historical data. Modern days in the software reliability field along with data support, machine and deep learning methods are used. The data that is defected is through the limitation of software test data. Although such defect data are relatively complete, problems, such as deficient amount, deficient coverage, and deficient data information have become the barrier of software reliability research and application [6].Software quality can be echoed by its non-functional requirements. The software quality depends mainly on reliability and efficiency. A good quality software should have less faults. The software reliability is calculated by the number of bugs within the software whereas software maintainability depends on how the software system should be maintained [7]. The main idea in software bug prediction depends on how effectively the software resources can be allocated. Software bug prediction models are designed to identify the defective modules. The performance measures are also used to improve the learning quality of the different classifiers. Different Machine Learning techniques provide different accuracies and these prediction accuracies are also dependent on the metrics for evaluations. Different strategies like feature selection techniques are also used to improve the performance [8].

II. RELATED WORK

From Fig. 1 we observe that, the overall percentage of papers on bug prediction in general and the actual percentage of papers on bug prediction using Machine Learning has gradually been increasing over the years.

Various techniques have been implemented on bug prediction and these techniques has given a feasible solution for the topics like what kind of metrics to be used for analysis, how to identify whether the attributes have bug or not, whether the data collection should be from equivalent project and so on [19]. There is evidence that bugs can be predicted before their detection.

By collecting few preceding versions of data, we could be able to predict bugs with feasible accuracy. Machine Learning is one such way that can be used for making necessary predictions and it is important because it works fine for several huge number of bug datasets [20]. The following are some of the existing methods discussed and it provides the information about the accuracy obtained.

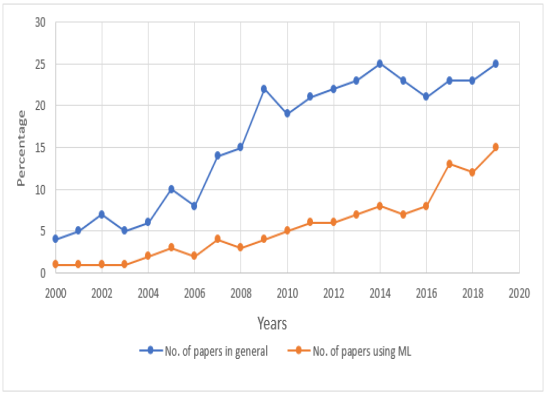

Fig. 1: Statistics from Scopus: The overall percentage of papers on bug prediction in general over actual papers on bug prediction using machine learning.

A. By the method of clustering

An experiment was performed on the SBP model based on the clusters to improve the performance. Once the method is applied the researcher updates the performance from 31.6 % to 79.2% recall and precision from 65.8 % to 78.6%. The bug proneness of object-oriented programming by the application of k-mean based clustering method was found to be 62.4% accurate [2].

B. By the method of Ensemble

Model building process use ensemble method. The aim of this method was to predict a bug. The dataset was divided into method, class and package levels. NASA KC1 dataset was used for method and class levels and eclipse dataset was used for package level with the help of ensemble methods. It is identified that bagging ensemble method have a better performance for method and package data. Method level results are as follows bagging is 0.809, boosting is 0.782, staking is 0.79 and voting is 0.63. Package level data was identified as follows bagging is 0.82, boosting is 0.78, staking is 0.72, and voting is 0.76. Class level metric, the results are as follows bagging is 0.78, boosting is 0.74, staking is 0.8 and voting is 0.82 [2].

C. By the method of Decision Tree

Decision Tree is a hierarchal and a model used for prediction which uses observations as branches to meet the resultant value in the leaf. Decision Tree constitutes decision nodes, the decision node itself consists of branch and leaf nodes that is nothing but a decision [9]. The model predicts 73% accuracy.

D. By the method of object-oriented metrics

Object oriented metrics method is a collection of defined data and predefined operations that are performed on data. This method has become so popular such that it provides a best way of analyzing a problem, provides a design for a solution and implements it. The advantage of this method is that it can be reused, can be more reliable and can be maintained easily than any other methods. How this method is useful in detecting the bug is carried out through describing the data by collecting similar kinds of project. Then assumption of data which is done by comparing the defect dataset with the instance of the corresponding dataset. Once the data assumption is done validation of data is carried out just to ignore the duplicate entries. Description of metrics is classified into dependent which is a bug and independent which includes metrics. The assumption of metrics is then done for better development of software bug prediction model and thus the validation of metrics is done by selecting proper suitable metrics for bug prediction. The model predicts 76.27% accuracy [10].

E. By the method of Naive Bayes (NB)

Detection of bug through Naïve Bayes is carried out through collection of some well-known projects out of which top five bugs will be used to train the classifiers. So overall, we can see an accuracy of 75% once the proposed model is developed. Here the accuracy is mainly dependent on the number of bugs that has been performed on the project which have been selected [11]. Bayesian Networks are used for predicting software quality in three ways: the methods used for learning parameters, structure, and type of variables that represent Bayesian Network nodes. Variables in Bayesian Networks are considered as categorical. Bayesian Networks is used less due to high dependency on knowledge and resources [12].

F. By the method of Support Vector Machine (SVM)

Support vector machine is for classifying the set of input parameters into one of the several classes for prediction in number of bugs. To classify the input parameters into different class's multi class support vector machine is proposed. The project was carried on ant 1.7 dataset. Each record provides one output parameter and rest of them are metrics [13]. The model predicts 78% accuracy.

G. By the method of KNN

KNN is used for software bug prediction. In order to find, to which category does the new data point falls the method follows Euclidian distance and Manhattan distance. When this method was used on xerces-1.3 dataset it was observed that the accuracy was around 76.82% [14].

H. By the method of Random Forest Classification

For the dataset that has been collected it designs several decision trees and by using entropy index and Gini index we get prediction result. This also choses the best decision tree along with the expected result. When this method was used on xerces-1.2 dataset we observed that the model predicted 78.39% accuracy [14]. Bagging or Bootstrap aggregation is a method which reduces the variation of prediction by combining the results of more than one classifier of a same data set that works on variety of sub-samples. Consistent performance can be observed.

Retrieval Number: B4119079220/2020©BEIESP
DOI:10.35940/ijrte.B4119.079220
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication

I. By the method of Hyper parameter Optimization

Here we are going to improvise the prediction of bug by optimizing the hyper parameters of a machine learning model. These methods are carried out on five projects. The results what was observed for each scenario was, break the datasets into tuning set and experimentation set. The tuning of hyper parameters is done by tuning set. For machine learning model, prediction error is compared between the default hyper parameter values and tuned values. Comparison is done through cross-validation. Student's t-test is utilized for comparison and identifies whether the tuning process helps in better accuracy or not [15]. Also, it is observed that the results showed by hyper-parameter optimization using Artificial Immune Network performs better than the default hyper-parameters used. The model predicts 73.9% accuracy [16].

J. By the method of Artificial Neural Network

A complex relationship is portrayed between an input and an output, as ANN is a nonlinear classifier. A result is produced by performing all together by the processing units consisting in neural network [17]. The links between the neurons sends a signal to another neurons and each helps to evaluate the result using an activation function by summing up all neuron's inputs. The model predicts 79% accuracy.

K. By the method of Logistic Regression

Logistic regression algorithm is a method used for predicting a bug in a software. Here we are considering Logistic Regression as a base model and others as benchmarking models. We validate the model performance to identify the accuracy, confusion matrix and classification report. With these measures we have been trying to improvise the model performance and predict the bug. The model predicts 77% accuracy. The Logistic Regression (LR) depends on how we analyze a problem, results are measured with the help of values such as 0 (true) and 1 (false). The goal of logistic Regression is to find the perfect model to describe relationship between software quality project and test management data [18].

L. By the method of Convolutional Neural Network

Bug prediction using Convolutional Neural Network provides weightage in deep learning for adequate feature extraction. With the help of this technique we can obtain good accuracy [21]. In this methodology, firstly the program of train and test sets are parsed into abstract syntax trees. The nodes are then selected from each tree and are converted into token vectors. These token vectors are fed to the encoding phase where they are encoded into numerical vectors and are fed to CNN. Convolutional Neural Network generates semantic and structural features for program and these features are combined with the existing bug prediction features. The features are then given as an input to logistic regression model to conclude whether the program has bug or not [22].

Table- I: Review of different ML methods applied on bug detection

Method	Accuracy	Pros	Cons
Clustering [2]	62.4%	Resource availability is more and if one cluster fails another cluster can take up the work.	It is expensive since cluster requires better hardware and design.
Ensemble Method [2]	78%	It is fast, simple and easy to code. It can be used for the data like numeric, discrete or textual.	It is expensive and memory constraints are present.
Decision Tree [9]	73%	It does not take much effort for data preparation during pre-processing and it does not need normalization and scaling of data.	Modification of the data can affect structure of the decision tree. Complexity and time taken is more.
Object-Oriented Metrics [10]	76.27%	Considering the use of earlier approach based on functional decomposition, Object-Oriented Metrics provides better reusability, maintainability & reliability.	It is seen that most of the metrics depend on source code changes and these changes are not dependent on the software process used.
Naïve Bayes Classifier [11,12]	75%	Naïve Bayes is faster in predicting test data set, also it holds well for categorical than numerical variables.	If the categorical variable is not present in training dataset the model assigns zero probability which will be unable to predict.
Support Vector Machine [13]	78%	For high dimensional spaces it is accurate and memory efficient.	For larger dataset it is not much efficient. It is susceptible to overfitting.
K-Nearest Neighbor [14]	76.82%	It is used for both classification and regression. It does not need any assumptions and is very simple.	If the dataset is large than speed of the algorithm slows down. It is not good for imbalanced data.
Random Forest Classification [14]	78.39%	It has the capability to handle large datasets with higher dimensionality. It can estimate the missing data and maintain the accuracy when large data is missed.	It is not suitable for regression problem as it does not predict the training data and it might over fit the data.
Hyper parameter Optimization [15,16]	73.9%	It can be easily parallelized.	It is computationally expensive.

Artificial Neural Network [17]	79%	It is flexible and can be used for both classification and regression problems. Any data can be made as numeric. It is reliable for the tasks involving more features. The predictions are faster once trained.	It is costly and more time is required to train. Since it deals more with training data, problems can be seen such as overfitting and generalization.
Logistic Regression [18]	77%	It is easy to implement and efficient to train, also we get the conditional probabilities free.	The hypothesis space is limited. Non-linear problems cannot be solved.
Convolutional Neural Network [21, 22]	61%	It is efficient and automatically extracts the important features.	It does not encode location and direction of objects into their prediction.

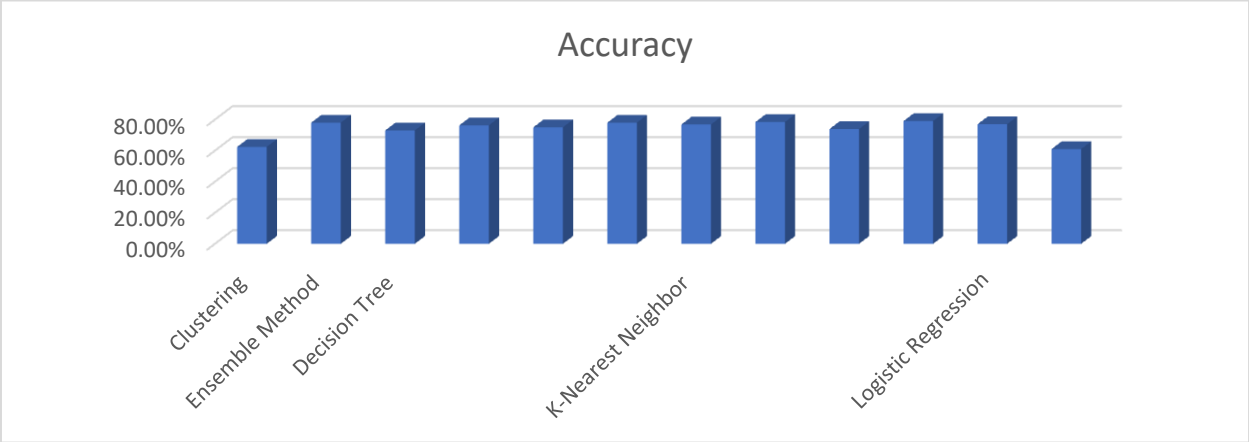

Fig. 2: Results of methods over accuracy.

III. CONCLUSION

The performance of software bug prediction presented in this article has been taken from real-time applications. Software bug prediction is implemented in order to find some unique characteristics and also to improve the quality of the developed software. It identifies the modules which has bugs in it. Overall, the article provides the analysis about the various results of the research taken from literature overview of machine learning in software bug prediction. Thus, the results of these existing methods show how it yields the accuracy when compared to other existing algorithms. Still efforts have been put to explore more on this methodology for predicting a bug. Moreover, by incorporating Artificial Neural Network (ANN) technique which helps in improvising the accuracy for large datasets, along with various machine learning algorithms. This would help in designing and development of a model that predicts the bug in a software, which would be effective enough to be used productively by the software industries.

REFERENCES

1. S. Delphine Immaculate, M. Farida Begam and M. Floramary, "Software Bug Prediction Using Supervised Machine Learning Algorithms," 2019 International Conference on Data Science and Communication (IconDSC), Bangalore, India, 2019, pp. 1-7, doi: 10.1109/IconDSC.2019.8816965.
2. Feidu Akmel, Ermiyas Birihanu, Bahir Siraj "A Literature Review Study of Software Defect Prediction using Machine Learning Techniques," IJERMT, ISSN: 2278-9359 (Volume-6, Issue-6), June 2017.
3. Dr. R Beena, N. Kalaivani, "Overview of Software Defect Prediction using Machine Learning Algorithms," International Journal of Pure and Applied Mathematics Volume 118 No. 20 (2018), 3863-3873, ISSN: 1314-3395.
4. S. Ray, "A Quick Review of Machine Learning Algorithms," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019, pp. 35-39, doi: 10.1109/COMITCon.2019.8862451.
5. J. Gao, L. Zhang, F. Zhao and Y. Zhai, "Research on Software Defect Classification," 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 2019, pp. 748-754, doi: 10.1109/ITNEC.2019.8729440.
6. J. Xu, L. Yan, F. Wang and J. Ai, "A GitHub-Based Data Collection Method for Software Defect Prediction," 2019 6th International Conference on Dependable Systems and Their Applications (DSA), Harbin, China, 2020, pp. 100-108, doi: 10.1109/DSA.2019.00020.
7. S. Reddivari and J. Raman, "Software Quality Prediction: An Investigation Based on Machine Learning," 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), Los Angeles, CA, USA, 2019, pp. 115-122, doi: 10.1109/IRI.2019.00030.
8. P. Singh, "Learning from Software defect datasets," 2019 5th International Conference on Signal Processing, Computing and Control (ISPCC), Solan, India, 2019, pp. 58-63, doi: 10.1109/ISPCC48220.2019.8988366.
9. Awni Hammouri, Mustafa Hammad, Mohammad Alnabhan, Fatima Alsarayrah, "Software Bug Prediction using Machine Learning Approach," IJACSA, Vol. 9, No. 2, 2018.
10. Gupta, D.L., Saxena, K. "Software bug prediction using object-oriented metrics," Sādhanā 42, 655–669 (2017). https://doi.org/10.1007/s12046-017-0629-5.
11. M. Sujon, M. Shafiuzzaman, M. M. Rahman and R. Rahman, "Characterization and localization of performance-bugs using Naive Bayes approach," 2016 5th International Conference on Informatics, Electronics and Vision (ICIEV), Dhaka, 2016, pp. 791-796, doi: 10.1109/ICIEV.2016.7760110.
12. Ayse Tosun, Ayse Bener, "A Mapping Study on the Bayesian Networks for Software Quality Prediction," Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, 2014.

13. L. Bao, X. Xia, D. Lo and G. C. Murphy, "A Large Scale Study of Long-Time Contributor Prediction for GitHub Projects," in IEEE Transactions on Software Engineering, doi: 10.1109/TSE.2019.2918536.
14. S. Agarwal, S. Gupta, R. Aggarwal, S. Maheshwari, L. Goel and S. Gupta, "Substantiation of Software Defect Prediction using Statistical Learning: An Empirical Study," 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), Ghaziabad, India, 2019, pp. 1-6, doi: 10.1109/IoT-SIU.2019.8777507.
15. H. Osman, M. Ghafari and O. Nierstrasz, "Hyperparameter optimization to improve bug prediction accuracy," 2017 IEEE Workshop on Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), Klagenfurt, 2017, pp. 33-38, doi: 10.1109/MALTESQUE.2017.7882014.
16. Sultan Alamri, Bushra Mumtaz, F Faiza Khan, Summrina Kanwal, “Hyper-Parameter Optimization of Classifers, Using an Artificial Immune Network and Its Application to Software Bug Prediction,” IEEE Access (Volume: 8), 2020.
17. K. K. Sabor, M. Nayrolles, A. Trabelsi and A. Hamou-Lhadj, "An Approach for Predicting Bug Report Fields Using a Neural Network Learning Model," 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Memphis, TN, 2018, pp. 232-236, doi: 10.1109/ISSREW.2018.00011.
18. Y. Zhou and J. Yan, "A Logistic Regression Based Approach for Software Test Management," 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Chengdu, 2016, pp. 268-271, doi: 10.1109/CyberC.2016.59.
19. shaniArora, VivekTetarwal, AnjuSaha, “Open Issues found in Software Defect Prediction,” Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace & Island Resort, Kochi, India
20. Subbiah, U and Ramachandran, M and Mahmood, Z (2019) “Software engineering approach to bug prediction models using machine learning as a service (MLaaS),” In: ICSOFT 2018 - 13th International Conference on Software Technologies, 26th to 28th July 2018, Porto, Portugal.
21. Amruthalakshmi, Prashanth Kambli, Naresh E, “Recognition of Handwritten Digits Using Convolutional Neural Network and Linear Binary Pattern,” International Journal of Innovative Technology and Exploring Engineering, ISSN: 2278-3075, Volume-9, Issue-1 November 2019.
22. J. Li, P. He, J. Zhu and M. R. Lyu, "Software Defect Prediction via Convolutional Neural Network," 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), Prague, 2017, pp. 318-328, doi: 10.1109/QRS.2017.42.

AUTHORS PROFILE

Prashanth Kambli is a M.Sc. (Engineering) by research degree holder and is serving as an assistant professor in IS&E department of MSRIT. Membership of professional societies: ISTE, IEEE, ACM.

Rashmi P is completed her bachelor degree in the stream of CSE and now pursuing her Master degree in Software Engineering.