PREDICTION OF SOFTWARE BUGS USING SUPERVISED MACHINE LEARNING TECHNIQUES

[1]P.Elango, [2]R.Sakthi, [3]S.Senthilvelan
[1] II MCA, Paavai Engineering College, Namakkal
[2]II MCA, Paavai Engineering College, Namakkal
[3] Professor, Department of MCA, Paavai Engineering College, Namakkal

Abstract: Software Bug Prediction is a critical topic in the software development and maintenance lifecycles that affects the overall success of software. This is so that software quality, reliability, efficiency, and cost can all be improved by foreseeing issues in advance. However, creating a reliable bug prediction model is a difficult endeavor, and numerous methods have been put forth in the literature. In this study, a machine learning (ML)-based prediction model for software bugs is presented. Based on past data, three supervised ML systems have been used to forecast potential software flaws. These classifiers include AdaBoost, decision trees, and Bagging. The evaluation method demonstrated that ML algorithms may be applied successfully and accurately. In order to evaluate the suggested prediction model with other strategies, a comparison measure is also used. The gathered data indicated that the ML technique performs better.

Keywords: Machine Learning (ML), AdaBoost, decision trees, and Bagging

I. INTRODUCTION

The presence of programming bugs influences decisively on programming dependability, quality and support cost. Accomplishing without bug programming additionally is difficult work, even the product applied cautiously in light of the fact that most time there is covered up bugs. As well as, creating programming bug forecast model which could foresee the broken modules in the beginning stage is a genuine test in programming.

Programming bug expectation is a fundamental movement in programming improvement. This is on the grounds that anticipating the buggy modules preceding programming organization accomplishes the client fulfillment, further develops the general programming execution. In addition, foreseeing the product bug early further develops programming variation to various conditions and builds the asset use.

Different strategies have been proposed to handle Software Bug Prediction (SBP) issue. The most realized methods are Machine Learning (ML) procedures. The ML methods are utilized broadly in SBP to anticipate the buggy modules in light of authentic issue information, fundamental measurements and different programming processing strategies.

In this paper, three regulated ML learning classifiers are utilized to assess the ML abilities in SBP. The review examined AdaBoost classifier, Decision Tree (DT) classifier and Bagging classifier. The talked about ML classifiers are applied to three distinct datasets got from [1] and [2] works.

The remainder of this paper is coordinated as follow. Segment 2 presents a conversation of the connected work in SBP. An outline of the chose ML calculations is introduced in Section 3. Area 4 depicts the datasets and the assessment procedure. Trial results are displayed in Section 5 followed by ends and future works.

II. RELATED WORK

There are many examinations about programming bug forecast utilizing AI methods. For instance, the concentrate in [2] proposed a direct Auto-Regression (AR) way to deal with anticipate the flawed modules. The review predicts the product future shortcomings relying upon the authentic information of the product amassed deficiencies. The concentrate likewise assessed and contrasted the AR model and the Known power model (POWM) utilized Root

Mean Square Error (RMSE) measure. Notwithstanding, the review utilized three datasets for assessment and the outcomes were promising.

The examinations in [3], [4] broke down the materialness of different ML techniques for issue expectation. Sharma and Chandra [3] added to their review the most significant past explores about every ML methods and the latest things in programming bug forecast utilizing Machine Learning. This study can be utilized as ground or move toward plan for future work in programming bug expectation.

R. Malhotra in [5] introduced a decent deliberate survey for programming bug expectation strategies, which utilizing Machine Learning (ML). The paper incorporated a survey of the multitude of concentrates between the time of 1991 and 2013, dissected the ML strategies for programming bug expectation models, and evaluated their presentation, looked at among ML and measurement methods, looked at between changed ML procedures and summed up the strength and the shortcoming of the ML procedures.

In [6], the paper gave a benchmark to permit to normal and valuable correlation between various bug forecast draws near. The review introduced a complete correlation between a notable bug forecast draws near, likewise presented new methodology and assessed its presentation by building a decent examination with different methodologies utilizing the introduced benchmark.

D. L. Gupta and K. Saxena [7] fostered a model for object-situated Software Bug Prediction System (SBPS). The review consolidated comparable kinds of imperfection datasets which are accessible at Promise Software Engineering Repository. The review assessed the proposed model by utilizing the exhibition measure (exactness). At last, the review results showed that the typical proposed model precision is 76.27%.

Rosli et al. [8] introduced an application involving the hereditary calculation for issue inclination expectation. The application gets its qualities, for example, the item arranged measurements and count measurements values from an open source programming project. The hereditary calculation involves the application's qualities as contributions to produce rules which utilized to arrange the product modules to inadequate and non-deficient modules. At long last, envision the results utilizing hereditary calculation applet.

The concentrate in [9] surveyed different article situated measurements by utilized AI methods (choice tree and brain organizations) and factual procedures (sensible and direct relapse). The aftereffects of the review showed that the Coupling Between Object (CBO) metric is the best measurement to foresee the bugs in the class and the Line Of Code (LOC) is genuinely well, yet the Depth of Inheritance Tree (DIT) and Number Of Children (NOC) are untrusted measurements.

Singh and Chug [10] talked about five well known ML calculations utilized for programming deformity expectation for example Counterfeit Neural Networks (ANNs), Particle Swarm Optimization (PSO), Decision Tree (DT), Naïve Bayes (NB) and Linear Classifiers (LC). The review introduced significant outcomes including that the ANN has least mistake rate followed by DT, however the straight classifier is superior to different calculations in term of imperfection forecast precision, the most well known techniques utilized in programming deformity expectation are: DT, BL, ANN, SVM, RBL and EA, and the normal measurements utilized in programming deformity expectation review are: Line Of Code (LOC) measurements, object arranged measurements, for example, union, coupling and legacy, additionally different measurements called mixture measurements which utilized both item situated and procedural measurements, besides the outcomes showed that most programming deformity forecast concentrated on utilized NASA dataset and PROMISE dataset.

Besides, the examinations in [11], [12] talked about different ML methods and gave the ML capacities in programming deformity forecast. The examinations helped the designer to involve valuable programming measurements and appropriate information mining method to improve the product quality. The concentrate in [12] decided the best measurements which are valuable in imperfection expectation like Response for class (ROC), Line of code (LOC) and Lack Of Coding Quality (LOCQ).

Bavisi et al. [13] introduced the most famous information mining method (k-Nearest Neighbors, Naïve Bayes, C-4.5 and Decision trees). The review broke down and analyzed four calculations and talked about the benefits and impediments of every calculation. The consequences of the review showed that there were various variables influencing the exactness of every method; like the idea of the issue, the utilized dataset and its exhibition framework.

The explores in [14], [15] introduced the connection between object-situated measurements and issue inclination of a class.

Singh et al. [14] showed that CBO, WMC, LOC, and RFC are powerful in foreseeing abandons, while Malhotra and Singh [15] showed that the AUC is compelling measurement and can be utilized to anticipate the flawed modules in beginning stages of programming advancement and to work on the exactness of ML strategies.

This paper examines three notable ML methods AdaBoost, Decision Tree and Bagging . The paper likewise assesses the ML classifiers utilizing different execution estimations (for example exactness, accuracy, review, F-measure and ROC bend). Three public datasets are utilized to assess the three ML classifiers.

Then again, the vast majority of the referenced related works examined more ML procedures and different datasets. A portion of the past investigations mostly cantered around the measurements that make the SBP as productive as could be expected, while other past examinations proposed various strategies to foresee programming bugs rather than ML procedures.

III. METHODOLOGY

Three supervised machine learning algorithms— AdaBoost, Decision Tree and Bagging —will be analysed and evaluated in this paper (DT). The paper includes a comparative examination of the chosen ML algorithms and demonstrates the performance accuracy and capacity of the ML algorithms in software bug prediction.

In order to be able to predict the output values for new input data based on the derived inferring function, supervised machine learning algorithms attempt to develop an inferring function by drawing conclusions about relationships and dependencies between the known inputs and outputs of the labelled training data. The selected supervised ML methods are briefly described below:

Decision Tree: Data mining frequently employs the decision tree (DT), a popular learning technique. When we talk about DT, we're talking about a hierarchical, predictive model that employs the item's observations as branches to get to the item's target value in the leaf. DT is a tree having leaf nodes that reflect the decision and decision nodes that have several branches.

Bagging: In order to increase the accuracy of unstable classification systems, Breiman [21] created bagging, along with bootstrap and aggregation approaches. For bagging, a decision tree is constructed using X samples and X bootstrap datasets with X randomly chosen examples and replacement from Y. By a majority vote, the expected new sample class is determined. The outcomes of comparing new cases to X decision trees are documented. Although a single decision tree's straightforward interpretation is lost, bagging increases the precision of categorization rules. In this investigation, decision stump and bagging with J48 are used[22]."Boosting" is used to increase a particular learning algorithm's accuracy. A machine learning technique called "boosting" identifies and combines loose rules to obtain precise categorization. The boosting technique uses a variety of training set subsets from the base learning to repeatedly find rules.

AdaBoost: The first effective boosting technique for binary classification was called AdaBoost, short for Adaptive Boosting. It is an algorithm for supervised machine learning and is used to improve the performance of all machine learning algorithms. Like decision trees, it works best with reluctant students. These are models whose categorization accuracy is slightly better than random chance.

IV. EVALUATION METHODOLOGY

Three separate datasets, DS1, DS2, and DS3, were techniques in this investigation. The two metrics that make up each dataset are the number of faults (Fi) and the number of test

employees (Ti) for each day (Di) over the course of a software project. 56 measurements from the DS1 dataset were used in the testing procedure described in [1]. DS2, which was also obtained from [1], measured system errors over the course of 100 consecutive days of testing a software system made up of 200 modules, each with one kilo line of Fortran code. DS2 includes 101 measures. The real measured data for a test/debug programme of a real-time control application provided in [18] is contained in DS3, which was produced in [2]. The datasets underwent preprocessing using a suggested clustering method. The proposed clustering algorithm assigns class labels to the data. These labels are designed to divide the total number of errors into the following five categories: A, B, C, D, and E. The value of each class and the number of examples that fall under it in each dataset are displayed in Table IV.

We techniques a set of well-known measures [19] based on the generated confusion matrixes to assess the effectiveness of utilizing ML algorithms in software bug prediction. The confusion matrix and the techniques evaluation metrics are described in the next subsections.

Table 1. NUMBER OF FAULTS CLASSIFICATION

Fault Class	Number of Faults	Number of Instances		
		DS1	DS2	DS3
A	0-7	35	80	65
B	8-14	5	32	44
C	15-21	4	10	8
D	22-28	6	12	19
E	More than 28	5	4	2

V. RESULTS

Weka, a machine learning tool, was utilized in this work to assess three ML algorithms (AdaBoost, DT, and Boosting) in the topic of software bug prediction. For each dataset, cross-validation (5-fold) is the technique.Table 2 displays the performance of the classifiers for the three datasets.

Table 2: Classification Accuracy achieved by Different Techniques

Data Set	AdaBoost	Baggin g	DT
DS1	0.88	0.91	0.86
DS2	0.92	0.96	0.92
DS3	0.95	0.98	0.94

VI. CONCLUSIONS

Software bug prediction is a technique that uses historical data to build a prediction model to forecast potential software flaws in the future. Different strategies have been put forth using various datasets, metrics, and performance measures. This research assessed the application of machine learning methods to the problem of software bug prediction. Three machine learning methods—DT,AdaBoost, and Bagging—have been techniques.Three actual testing/debugging datasets are used to carry out the evaluation process. On the basis of metrics for accuracy, precision, recall, F-measure, and RMSE, experimental findings are compiled. Results show that ML techniques are effective methods for anticipating software issues in the future. The comparison findings demonstrated that the Bagging classifier outperformed the others in terms of results.

REFERENCES

[1] Y. Tohman, K. Tokunaga, S. Nagase, and M. Y., “Structural approach to the estimation of the number of residual software faults based on the hyper-geometric districution model,” IEEE Trans. on Software Engineering, pp. 345–355, 1989.

[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.

[3] D. Sharma and P. Chandra, "Software Fault Prediction Using Machine-Learning Techniques," Smart Computing and Informatics. Springer, Singapore, 2018. 541-549.

[4] R. Malhotra, "Comparative analysis of

statistical and machine learning methods for predicting faulty modules," Applied Soft Computing 21, (2014): 286-297

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

[6] 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.

[7] Gupta, Dharmendra Lal, and Kavita Saxena. "Software bug prediction using object-oriented metrics." *Sādhanā*(2017): 1-15..

[8] M. M. Rosli, N. H. I. Teo, N. S. M. Yusop and N. S. Moham, "The Design of a Software Fault Prone Application Using Evolutionary Algorithm," IEEE Conference on Open Systems, 2011.

[9] T. Gyimothy, R. Ferenc and I. Siket, "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," IEEE Transactions On Software Engineering, 2005.

[10] Singh, Praman Deep, and Anuradha Chug. "Software defect prediction analysis using machine learning algorithms." 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence, IEEE, 2017.

[11] M. C. Prasad, L. Florence and A. Arya, "A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques," International Journal of Database Theory and Application, pp. 179-190, 2015.

[12] Okutan, Ahmet, and OlcayTanerYıldız. "Software defect prediction using Bayesian networks." Empirical Software Engineering 19.1 (2014): 154-181.

[13] Bavisi, Shrey, Jash Mehta, and Lynette Lopes. "A Comparative Study of Different Data Mining Algorithms." International Journal of Current Engineering and Technology 4.5 (2014).

[14] Y. Singh, A. Kaur and R. Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness models," Software Qual J, p. 3–35, 2010.

[15] Malhotra, Ruchika, and Yogesh Singh. "On the applicability of machine learning techniques for object oriented software fault prediction." Software Engineering: An International Journal 1.1 (2011): 24-37.

[16] A.TosunMisirli, A. se Ba¸ S.Bener,"A Mapping Study on Bayesian Networks for Software Quality Prediction", Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, (2014).

[17] T. Angel Thankachan1, K. Raimond2, "A Survey on Classification and Rule Extraction Techniques for Data mining",IOSR Journal of Computer Engineering ,vol. 8, no. 5,(2013), pp. 75-78.

[18] T. Minohara and Y. Tohma, "Parameter estimation of hyper-geometric distribution software reliability growth model by genetic algorithms", in Proceedings of the 6th International Symposium on Software Reliability Engineering, pp. 324–329, 1995.

[19] Olsen, David L. and Delen, " Advanced Data Mining Techniques ", Springer, 1st edition, page 138, ISBN 3-540-76016-1, Feb 2008.

[20] L. H. Crow, "Reliability for complex repairable systems," *Reliability and Biometry*, *SIAM*, pp. 379–410, 1974.

[21] Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140

[22] Sasi Kumar.M, S SenthilVelan(2020) "Enhanced Edge Based Image Retrieval Using Boosting Framework", International Journal Of Current Engineering And Scientific Research (IJCESR), Issn : 2394-0697, Vol. No7(7), 2020, Pp. 37-41.