IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Software Defect Prediction Stratagem

Ritik Tailor
Poornima Institute of Engineering and Technology
Jaipur, India

***Abstract*— Software Defect Proneness and its Prediction is a very important process in the software development life cycle which allows the developers and necessary stakeholders to identify the gap between the user and the use case. We can easily detect and rectify the errors and problems based on operation, function and credibility of the software, from modules to classes. Parts of the software which are prone to defect can be made more effective and resource allocation can be done very effectively to upgrade the overall efficiency of the software. There might exist cases when the product is not in an object oriented metric and the stakeholders might face serious ethical or technical issues with the software, hence making it neutralized and de-escalated. The main anchor of the review is to find the well known and efficient methods of software fault prediction and the advantages and disadvantages associated with them. A taxonomical classification of all the studied methods is provided. The paper is concluded with observations, learnings, challenges and directions for the future.**

Keywords—Software Fault Prediction, software metrics, defect analysis, software defect prediction.

I. INTRODUCTION

Software fault prediction helps program quality assurance teams to optimize their assets in-between software reliability checks. Recently, several threshold-based approaches have been proposed in literature. With an ever increasing complexity of software services and products, software fault prediction has become a very determining process in software quality assurance or software fault detection. In the early stages of software development, the developers and the stakeholders made very simple means to check for any ambiguity in the software products or services. Methods like LOC or Lines of Codes and cyclomatic complexity were used exhaustively to enumerate the defects of a software.

Since times we have been developing models and different methods to study the defect and its prediction. Modern software architecture has become very complex and it is not possible for us to run the analysis on the whole of the software, as it is a very time taking and a budget breaking process. Testing like this has always been carried out in a time constraint and most of the time, and with a very limited number of resources. A lot of software metrics are now being considered in various models. A software metric is a finite and countable characteristic of a software which can be considered for analyzing software performance. There are two types - Process metrics and Product metrics. Product metrics are the characteristics of the product itself and the latter are the characteristics of the processes that are essential from the development to the deployment itself. Some examples of metrics could be - LOC, class level metrics and function metrics.

There are two main aspects to consider for any method - the number of faults in the software and the impact of theirs on the software. The aim is always to reduce both these aspects but practically, on ground level, it is practically impossible to reduce both of them in an easy manner. We have to work in accordance and with relativity to the authority of the impact or the number of faults.

Even after these technological advancements, researchers are not being able to provide a generalized and a common subset of characteristics which can act as an 'always non faulty' module. There are no standard reporting measures to capture the impact of any inconsistency.

II. BACKGROUND

A. *Approach*

In the Early times, the main focus was on within-project Defect Prediction models which comprised working the data set and characteristics of the software itself. The empirical data, data set and its evaluation, accuracy analysis and efficiency analysis were done on that particular software only. Later, as the models were being developed, a new approach called cross project defect prediction was introduced and slowly came into practice.

In the early 1990's, the main approach that was used to find faults was classification and logistics regression. Python came out in 1991, as a successor of ABC programming language, so only the basic features of python could be used. AS with the developments, in the 2000's, Support Vector machines, Tree based machine learning models and change classification came into the spotlight as they were much faster and consumed less human effort and we could now automate them too.

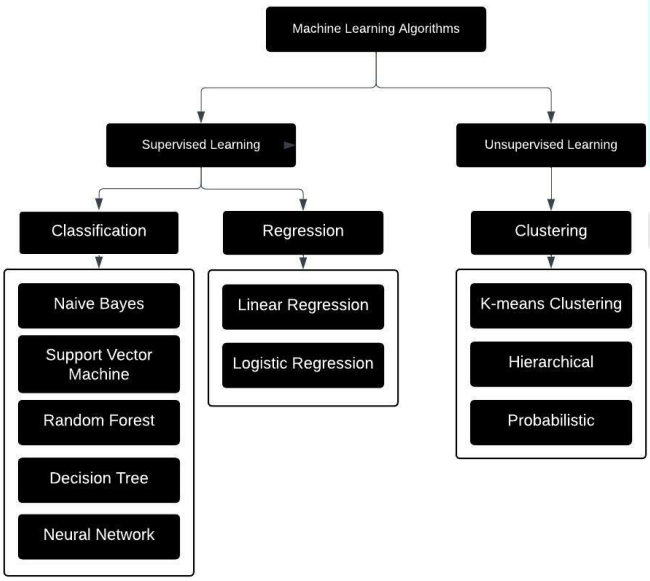
After 2010 and till now, we have evolved from using Neural Networks, Naive Bayes, Decision Trees to the present day Just In Time models. These just in time models can predict if there is a defect present in the software at every commit.

B. Role of machine learning in determination of fault

There are dozens of algorithms and their combinations which are used in determination and optimization of fault. A table of machine learning method and one its specific example is highlighted here:

	Basis	Technique
1	Tree based technique	ID3, CART
2	Perception Based	ANN
3	Statistical Method	Regression
4	Evolutionary Based	Genetic Algorithms
5	Kernel Based	Support Vector Machine
6	Bayesian Based	Naive Bayes
7	Ensemble Based	Random Forest
8	Instance Based	k-means

Most of the models based on machine learning work only on labeled datasets only. For any unlabeled dataset, Rakesh Kumar, Amrita Chaturvedi and Lakshamanan Kailashan introduced the TCLP approach. TCLP (Threshold clustering labeling plus) where it can easily identify the unlabeled datasets by self learning.



III. STRATAGEM

For any prediction model, there is a procedure to be followed for prediction.

A. Threshold Derivation

Thresholds are heuristic qualities that are utilized to fix scopes of desirable and undesirable metric qualities for considering the estimated software, apart from that, is used to recognise abnormalities that might become a genuine issue. Threshold values make sense in the event that a metric is in the normal range.

Threshold computation is the midmost of analyzed heuristic. Threshold derivation plays a supreme function for clustering and labeling the data points in our approach. Hence, the first stage is to decide the threshold of the software criteria . It's a well- known fact that software metrics are used to describe the internal quality of software code. This internal dimension of software quality assists both the programme developer and the tester in improving those quality characteristics for which the measure values are insufficient. As a result, software metrics may be used as a software quality evaluation procurator. Furthermore, high-complexity software legislation is undesirable since it is thought to be more fault-prone.

There are many methodologies accessible to figure out threshold values.

(1) It should be benchmark data

(2) Its analysis should be statistical

(3) It can be repeated.

B. Cluster Labeling

Hierarchical clustering segments a document assortment into few clusters, and each cluster is further divided into sub clusters in a recursive way. Hierarchical clusters can be developed by agglomerative styles that beginning with each document in its own cluster and furthermore continually bunch similar to clusters into more extensive clusters; or by troublesome styles that beginning with all documents in a single cluster and furthermore continually partition each cluster into more point by point subclusters. In labeling hierarchical clusters, bone expects the reality of a size of document clusters. The errand is to relegate a decent descriptor to each cluster tie in the scale. A list of terms is habitually less valuable than a single request marker, since it requires the stoner to deduce the origination induced by the terms. In any case, a rundown of terms is the most well-known decision for labeling clusters consequently on the grounds that it flops effortlessly; an individual can Much of the time derive the overall depiction for sure when a significant number of the named terms are unfortunate decisions.

C. Metric Selection

Software metrics acquired are typically linked to problems evaluated during pre-discharge and post-discharge. These software measurements and deficiency information are used to build software fault forecast algorithms. In this approach, the nature of currently in-process programme portions is analyzed, e.g., fault vulnerable or fault non-vulnerable. These strategies help to reduce software flaws and deliver highly strong products. Software defect forecasting systems have been a focus of research in the software design community. Defect forecasts are typically used by software quality check stakeholders to guide them towards limited projects sensibly resources towards programme parts that are likely to have

low reliability additionally, dependability. A real-world test that stakeholders consider is the sorting and selection of the suitable software metrics influenced structure and defect indication. Choosing the best static code metrics before building a defect prediction model offers several advantages, such as avoiding obnoxious features, reducing the arrangement of software metrics to use, or, in any case, further enhancing defect expectation execution. In many circumstances, feature (a metric of software products) selection techniques are efficiently utilized to set into a fraction of the set consisting of the most important attributes along with and characteristics.

D. Learning and Prediction

This is the last but not the least step of software defect prediction. The developed model is finally tested here with the results that it lays. Any efficiency or optimisation of the model and the software itself is carried out here only.

RELATED WORKS

Researchers and practitioners have been looking for new ways and combining already existing models with the new ones. A lot of research is being carried out on machine learning models. Machine learning models have been successful in less compromise and more efficiency of the results.

However, one literary work, that is based on the CLAMI technique, lays emphasis on an improvised technique. Most of the models work on labeled datasets only, but the improvisation works on TCLP. Threshold Clustering Labeling Plus works on random forest algorithms. Once the model is up and running, it can self learn and classify the artifacts into binary classes as - faulty and non faulty

Other literary works include genetic machine learning models which can help determine fault and defect at very initial stages of development. It uses object oriented metrics as data and then develops a learning classifier system LCS which has an IF and THEN state to learn and classify the metrics.

Another considerable literature is based on the k-means mechanism which works upon the PROMISE repository. K-means faces local optimum solution, and it requires multiple cluster center initialisations.

CONCLUSION

To forecast defects before real testing and reduce mistakes for the cost of expense and time of software products, the need for improved and more developed prediction methodologies will always be a constant source of interest. One's goal will always be to investigate the ease of access and to efficiently make use of genetic-based ML techniques for determining fault capacity in softwares using metrics, which are object oriented. We see areas of strength for any of laid out metrics with size measures, similar to lines of code complexity metrics and thus size measures appear to have a specific measure of predictive capacities. (Object-oriented) metrics perform better compared to intricacy and size metrics when thought about for predicting issues and defects. However showing a specific correlation to measure they include a few extra properties too. Static code metrics, similarly as complexity, size and object oriented metrics, are appropriate for noticing a specific version of a software, however with a dropping accuracy with each software cycle. Subsequently, they are not appropriate for profoundly iterative, post-release software, where the primary driver of faults is a result of the improvement cycle and not really due to the properties of size and design.

REFERENCES

[1] Amod Kumar, Ashwani Bansal "Software Fault Proneness prediction using genetic based machine learning" ,978-1-7281-1253-4/19 © 2019 IEEE

[2] Salim Moudache, Mourad BADRI "Software Fault Prediction BAsed on Fault Probability and Impact" , 2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA)

[3] Liu Xi, Li Haifeng, Xie Xuyang "Intelligent Radar Software Defect Prediction Approach and Its Applications",2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)and Applications (ICMLA)

[4] Rakesh Kumar, Amrita Chaturvedi and Lakshmanan Kailasam "An Unsupervised Software Fault Prediction Approach Using Threshold Derivation", IEEE TRANSACTIONS ON RELIABILITY, VOL 71,NO. 2, JUNE 2022

[5] R. Jothi "A comparative study of unsupervised learning algorithms for software fault prediction" Proceedings of the Second International Conference on Intelligent Computing and Control Systems (ICICCS 2018) CFP18K74-ART ISBN:978-1-5386-2842-3

[6] Mahesh Kumar Thota, Francis H Shajin, P. Rajesh "Survey on software defect prediction techniques" IJASE.202012_17(4).331

[7] Tina Beranic and Marjan Hericko " Approaches for Software Metrics Threshold Derivation: A Preliminary Review" SQAMIA 2017, 6th Workshop of Software Quality, Analysis, Monitoring, Improvement and Applications 11-13.9.2017

[8] Safa Omri and Carsten Sinz " Deep Learning for Software Defect Prediction: A Survey" ICSEW 20 42nd Conference on Software Engineering Workshop

[9] Wei Zheng, Tianren Shen, Xiang Chen and Peiran Deng "Interpretability application of the Just-in-Time software defect prediction model" Journal of Systems and Software, 2022

[10] Ishani Arora, Vivek Tetarwal and Anju Saha "Open Issues in Software Defect Prediction" ICICT 2014, Procedia Computer Science 46 (2015) 906-912

[11] N.C Shrikanth, Suvodeep Majumdar and Tim Menzies "Early Life Cycle Software Defect Prediction. Why? How?" arXiv:2011.13071v3 [cs.SE] 9 Feb 2021

[12] Anil Kumar Pandey and Manjari Gupta "Software Metrics Selection for Fault Prediction: A Review" International Journal of Management, TEchnology and Engineering ISSN NO: 2249-7455

[13] Jayanthi.R, Lilly Florence and Arti Arya "A review on Software Defect Prediction Techniques using Product Metrics" International Journal of Databases theory and application Vol.10 no.1 2017, pp. 163-174

[14] Huanjing Wang, Taghi M. Khoshgoftaar and Amri Naploitano "Choosing the Best Classification Performance Metric for Wrapper-based Software Metric Selection for Defect Prediction"