International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 12, Issue 01, January 2023)

An AI Model Based Defect Prediction in Software

Kishore Kumar Mishra[1], Pawan Agarwal[2]
[1,2]*Deparment of CSE, Compucom Institute of Technology and Management, Jaipur*

***Abstract*— One of the crucial components of software is software quality. Software design complexity is rising in response to rising demand, increasing the likelihood of software flaws. Tester improve the quality of software by fixing defects. Consequently, the study of flaws greatly raises the quality of software. The complexity of software also contributes to a greater number of flaws, making human detection a laborious procedure. The goal of this project is to create methods for the automatic detection of software flaws. In this research, we examine the performance of cutting-edge machine learning techniques for software defect classification. For this research, we used seven datasets from the NASA archive repository. Gradient Boosting and Neural Network classifiers outperform other methods in terms of performance.**

***Keywords*— Software Testing, Quality Assurance, Machine Learning, and NASA Promise Dataset.**

I.INTRODUCTION

The software industry is evolving quickly as a result of rising demand and technology. Defects are unavoidable in software since people design it for the most part. Defects can often be described as unwanted or unacceptable changes to software programmes, documents, and data[1].The product manager could misinterpret the needs of the customer during the requirements analysis, which could lead to a defect that persists throughout the system design stage. Inexperienced coders may also be to blame for errors in the code. Defects have a substantial impact on the quality of software, and they can have serious repercussions in the aerospace and health care industries. If the problem is found after deployment, it puts a strain on the development team because they will have to redesign some software components, which will raise the price of the project. Defects are a nightmare for well-known organizations. Customer dissatisfaction damages their reputation and, as a result, reduces their market share.

Software testing has consequently become one of the primary areas of concentration for findus trial research [2]. Additional manual methods take a lot of time and are ineffective as a result of the increase in software complexity and development. Automatic categorization of faults has been a popular topic in study due to the emergence ofmachine learning.

In this study, we first describe software defect details, including the numerous categories that are available in the literature, and then we discuss the manual categorization techniques that have been suggested by different academics. The analysis of the most advanced machine learning techniques for automatic software detection is presented last.

II. CONCEPTS AND OVERVIEW OF SOFTWARE DEFECTS

A.Concept of software defects

This article uses the IEEE729-1983 (Standard Glossary of Software Engineering Terminology) to characterise defects as, for example, programming faults, errors, and failures because analysts frequently find it difficult to discern between software defects and those. From the inside of the product, flaws are blunders and errors made during maintenance or product creation. A flaw, from an external perspective, is a breach of or a failure of the framework or system to carry out particular functions[3,4]. The list of ideas that are frequently confused with flaws is as follows.

1. *Fault:* The software runs in an internal state that is inappropriate and does not perform as expected by the client. We can think of it as a flaw that could lead to a software fault or disregard normal dynamic behaviour.
2. *Failure:* This is when the client rejects the outputs that the software generates while it is running. Consider this: if the framework is unable to satisfy the patch edasset's execution requirements, resulting in a loss of execution capacity and failure to meet client capabilities.
3. *Error:* It is introduced by people and transforms into fault under particular circumstances. It is present across the whole software life cycle, including in the design, data structure, code, requirements analysis, and other carriers of thesoftware[5].

The quantity of flaws determines the quality of software. A high number of flaws negatively impacted customer satisfaction, increased costs for the company, and slowed testing. Im demonstrating that test productivity is essential to managing faults in order to reduce expenses.

B. Main research directions of software defects

1. Managing software defects

The primary focus of defect management is the gathering, statistical analysis, and use-ful recording of faults.

Engineers have created numerous robotized defects management technologies to increase management productivity. Currently, the most widely used tools in the sector are Bugzilla, an open-source bug tracking framework offered by Mozilla, and JIRA, distributed by Atlassian. These two tools don't have a more professional-found examination or specific grouping of faults, but they do record the transactions, attributes, and statistici cal information of defects. Classification and defect analysis play a significant role in defect management. In order to examine and arrange abnormalities, it is necessary to further explore the data stored in JIRA and Bugzilla.

2. Investigation of software defects

Effect analysis is frequently used by software developers to assess programming and development quality. Software defect analysis is a tactic for defining perfection and identifying the causes of flaws. The goal of software defect analysis is to enable analysts to identify, track down, assess, and enhance test effectiveness. The three main categories of defects analysis methodologies are qualitative analysis, quantitative analysis, and attribute analysis [4]. Root Cause Analysis (RCA) and Software Fault Tree Analysis are two of the most common qualitative analysis techniques (SFTA). Single attribute analysis and multi-attribute analysis are the two most popular divisions of attribute analysis.

3. Classification of software defects

Different and intricate software flaws exist. The ability to organise and aggregate flaws more clearly can help programmers evaluate the quality of their work, increase the productivity of analysts, and alleviate the burden of analysis. The suggestion of repair techniques and reuse test situations can be aided by classification [2]. It can understand how flaws are distributed according to the Through categorization and analysis, software faults are prevented from occurring frequently, the software development cycle is greatly enhanced, and software quality is so increased [6,7]. Software defect analysis includes software defect classification as a key component. Defect classification is extremely important since the results directly affect the defect analysis process.

Software defect classification has previously been divided into two categories: programmed/automatic classification andhuman classification.

1. Manually classifying software flaws a software flaw Manual classification denotes that examiners classify flaws according to their knowledge of them. First, researchers established the optimal classification of faults. Based on their experience, they identify the flaw and type of defect. However, this strategy's classification cycle is very difficult and requires a huge crew. A lot of data analysis will result in a much slower classification speed than the computer because of limited human energy and memory, which will consume a lot of time and resources.
2. Automatic classification of software defects: To reduce development costs and improve development productivity, individuals are more inclined to use computers to automatically classify defects. Specialists are attempting to locate a straight forward method to classify defects, and the ascent of AI and machine learning has made the automatic classification of defects a hotspot for industrial research.

III. SOFTWARE DEFECTS MANUAL CLASSIFICATION METHOD

The quantitative analysis of faults is based on the classification of defects. Every individual distinguishes the defect type differently, for example, database type, code type, operation type, etc., due to the multiple components that produce flaws. Each category can continue to be subdivided; for instance, code categories could be divided into task errors, variable definition errors, etc. There are numerous classification techniques. The classification methods are also fairly complex because different classification techniques need be used for diverse analysis goals. Following are a few typical annual classification methods:

A. Orthogonal Defect Classification

Orthogonal Defect Classification (ODC), a different classification method, was introduced by T.J. Watson of I BM in 1992 [8]. IBM agrees that there should be no convergences, all defect classes should be orthogonal to one another, and each classification should be independent of the others. There shouldn't be a flaw that can occur with one categorization but also occur with another classification. Additionally, classes must completely encompass all problems, and there must be no defects that cannot be classified under any of the categories.

ODC identifies defects in detail, covers all phases of software development, and applies to any software product since the main goal of the organisation is to enhance the development process. ODC uses a variety of attributes to characterise the characteristics of defective elements. Eight fault attributes are defined in the most recent version [8]. The following eight classes of software flaws are determined by the characteristics of these traits: The following terms are used: assignment, verification, algorithm, timing, interface, function, association, and documentation.

B. Standard classification for Anomalies

The IEEE proposed the IEEE standard classification for anomalies [9] to provide software anomalies with a unified classification standard. The sestandard covers the entire software life cycle, effectively identifies, tracks, and optimises the cycle of development.

Failure classification and Defect classification are defined by the standard. Individuals must choose different attributes for classification, break down the classification findings, and determine whether to utilise the Failure classification method or the Defect classification technique initially before they can begin eclassifying Anomaly categorization can be divided into eight categories: computational issues, interface/timing issues, logical issues, data issues, data processing issues, document quality issues, documentation issues, and difficulties with cement. Each class can also be broken down further into more specific subclasses.This technological solution can provide relief for the project's requirements. As a result, the approach is more dependent on the executive's level of expertise. Individual subjective criteria, however, will significantly reduce the classification influence.

C. Thayer classification

Using error reports submitted by researchers during testing and client feedback, Thayer et al. categorise errors in their inclination [10,11]. This method categorises errors into global variable errors, requirement implementation errors, document errors, personnel operation errors, and logic, calculation, data processing, I/O, support software, operating system, interface, configuration, preset database, user demand changes, repeated, and errors. unidentified property error Additionally, they can bedivided into 164 subcategories.

This approach not only examines software problems but also system and employee operation errors.

It is widely and cautiously employed. The mistakes can be changed by developers, which is helpful. In any case, this method is unacceptable for streamlining the software development process because it doesn't take into account the causes of the issue.

D. Roger classification

Roger's taxonomy splits software flaws into 12 categories based on their root causes: intentional deviation specifications (IDS), insufficient or incorrect specifications (IES) [12]. Misunderstandings in customer communications (MCC), etc. The methodology analyses the reasons behind the method's and strategy's introduction since the classification strategy is typically simple, the defect information is scant, the standard is unreliable, and the analyzability isn't high.

E. Defect prevention classification of IBM

The IBM defect prevention classification method was proposed in 1990 by R. G. Mays and colleagues at the IBM Research Center. This method is similar to the STFA, which classifies flaws using causal analysis and iteratively until it is impossible to categorise them once more while taking the defect's introduction time into account. This approach divides defect categories into categories such as education, negligence, text errors, incorrect communication, etc. This method is alsosusceptible to abstract elements that result in a wide range of analytical findings for different people.

F. Putnam classification

It is incorrect to categorise all phases with coding errors classification models because the defects exist throughout the entire software development cycle, it is not comprehensive to consider just the errors in the coding stage, and the error attributes produced during each phase of requirement analysis and system design aredifferent. a result of this. Numerous faults that setf or wardthe classification method for Putnam's problems were examined by Putnam et al. [13]. Considering the diverse characteristics of numerous flaws throughout the development cycle and in accordance with when they first appeared, The flaws are divided into six categories: system design flaws, requirement analysis flaws, algorithm flaws, document flaws, performance flaws, and interface flaws. Using defect features, the approach groups defects into classification tree nodes. There is no code problem; this technique is only being taken into consideration during the development stage.

G.Michael classification

Michael presented two classification strategies[14]: classification of defect sgenerated by code and classification of modules in software projects—to evaluate the relationship between modules and defects. He thinks that modules and fault categories have a lot in common. For instance, interface modules are prone to interface flaws, while calculation modules are prone to variable mistakes, calculation problems, etc. This method of categorization divides defects into eight classes: computation type, data type, interface type, control logic type, platform type, user interface type, document type, and structure type. The strategy is accurate in classification, directly related to the module, and the logic is simple enough to assist the developer in identifying the flaw. Numerous analyses prove that the technique is a significant tactic for studying how to classify software defects[15].

Based on the optimizer's selection.

The most popular optimizer, which was also utilised in this experiment, is Adaptive Moment Estimation (ADAM) [25].

IV. Conclusion

Software flaws can negatively affect software quality, which can be problematic for both customers and developers. Manual software identification has become a challenging and time- consuming operation as software designs and technology have become more complicated. Consequently, during the past few years, industrial research on automatic software identification has picked up steam. In this study, we attempt to address this issue using machine learning and deep learning. We examine the outcomes of state-of-the- art machine learning methods using seven datasets from the NASA Promise dataset repository. There is still great room for development in this subject. In addition to thinking of fresh approaches that make use of sophisticated deep learning algorithms, academics should concentrate more on data gathering.

REFERENCES

[1] Y. Cai, Software reliability engineering foundation, Tsinghua university press,1995.

[2] J. Gao, L. Zhang, Z. Fengrong and Z. Ye, "Research on Software Classification," in Information Technology, Networking, Electronic and Automation Control Conference, 2019.

[3] I. C. Society, "IEEE 729-1983- IEEE Standard Glossary of Software Engineering Terminology,"1982.

[4] W. Bi, "Research on Software Defect Classification and Analy sis, "Computer Science, 2013.

[5] X. Yang and M. Duan,"Researcho f Software Defect Anal ysis Technology," Computer Engineering & Software, 2018.

[6] J. Collofello and B. P. Gosalla, "An application of causal analysis to the software modification process," Software: Practice and Experience, vol. 23, 1993.

[7] J. W. Horch, Practical Guideto Software Quality Manag ement, Artech House, 2003.

[8] R. Chillarege, I. Bhandari, J. Chaar, M. J. Halliday, D.S. Moebus, B. K. Ray and M.-Y. Wong, "Orthogonal Defect Classification - A Concept for In-Process Measurements, "IEEE Transactions on software Engineering, vol. 18, pp.943-956,1992.

[9] S. &. S. E. S. Committee, "IEEE 1044-1993 – IEEE Standard Classification for Software Anomalies, "IEEE, 1993.

[10] X. Huang, Softwarere liability, safety and quality assura nce, Electronic Industry Press, 2002.

[11] L. Meng-ren, "Research on Software Defects Classification, "Application Research of Computers, 2004.

[12] R. Pressman, Software engineering: apractitioner's approach, Palgrave Macmillan, 2005.

[13] L. Putnam and W. Myers, Measures for excellence: reliable software on time, within budget, Prentice Hall Professional Technical Reference,1991.

[14] I. Raphael and C. Michael, "Fault links: identifying module and faul types and their relationship, "2004.

[15] L. Macaulay, Human-computer interaction for software designers, Itp-Media, 1995.

[16] "NASA Promise Dataset Repository".

[17] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments, "Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol.374,2016.

[18] L. Breiman, "Random Forests," Machine Learning, vol.45,pp.5-32,2004.

[19] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," Nature, vol.521, pp.436-444,2015.

[20] "Logisticregression," Wikipedia.

[21] "Naïve Bayes," Wikipedia.

[22] "Gradient Boosting Classifier," Wikipedia.

[23] "Support Vector Machine," Wikipedia.

[24] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T.Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, M. Lungren and A. Ng, "CheX Net: Radiologist- Level Pneumonia Detection on Chest X-Rays with Deep Learning,"Arxiv, vol.abs/1711.05225, 2017.

[25] J. Baand D. P. Kingma, "Adam: A Method for Stochastic Optimization," Clinical Orthopaedics and Related Research, vol.abs/1412.6980,2015.