Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2020, 7(11):36-42

Research Article ISSN: 2394-658X

Machine Learning Methods for Predicting Software Bugs

Khirod Chandra Panda

Asurion Insurance, USA

Email id - khirodpanda4bank@gmail.com, Orcid id: 0009-0008-4992-3873

ABSTRACT

The objective of software bug prediction is to pinpoint which software modules are likely to contain bugs by leveraging basic project resources before actual testing commences. Early bug prediction is crucial due to the high costs associated with fixing bugs later in the process, particularly during the testing phase. Various techniques and methodologies, including widely used machine learning approaches, are employed to develop these prediction models due to their accuracy in delivering results and analyses. Consequently, I have conducted a review of existing literature on software bug prediction and machine learning to better comprehend the model construction process. My review not only examines the machine learning techniques utilized by previous researchers but also evaluates the datasets, metrics, and performance measures employed in model development. In my study, I analyzed 31 key studies and identified six distinct machine learning techniques. I found that two public datasets are commonly used, and that object-oriented metrics are frequently selected for building the prediction model. Performance is typically assessed using both graphical and numerical measures. My findings confirm that machine learning is effective in predicting bugs, although applications in this field are currently limited. The construction of prediction models presents several challenges, necessitating further research to achieve more definitive outcomes. Based on my findings, I also offer recommendations for future research in this area.

Keywords: Anomaly Detection; Data Mining; Time Series Data; Machine Learning Techniques.

1. INTRODUCTION

In the realm of software development, the concept of quality modeling is pivotal, as it directly influences the overall quality of the final product. This element of the development lifecycle is critical because it can significantly improve the quality of software by facilitating the early detection and resolution of bugs (Al-Jamimi, 2016). Early bug detection not only aids testers in the strategic allocation of resources but also enables the mitigation of bugs prior to software deployment (Xia et al., 2014), increasing the software's value while curbing costs, thus leading to more effective software management (Hassan et al., 2018). In today's rapidly evolving technological landscape, the complexity of software is accelerating, making testing an integral component of the development cycle. Menzies et al. (2010; Wahono, 2015) suggest that the likelihood of discovering bugs through predictive methods could surpass that of traditional review processes.

Consequently, the field of software bug prediction has become a hub of research activity, drawing scholars from various sectors who put forth a plethora of frameworks, models, and techniques for predicting software bugs. Efforts to refine these existing methods continue, even as the research field grapples with inherent uncertainties and each model comes with its own set of limitations. Machine learning emerges as a popular domain within this field, with several algorithms like neural networks, support vector machines, and Bayesian networks deployed for bug detection. Practitioners have access to a variety of public datasets, such as those from the PROMISE and NASA MDP repositories, which are enriched with metrics like Halstead and McCabe metrics. These tools are crucial for assessing the efficacy of proposed models through performance metrics such as the Area Under Curve (AUC) or F-Measure.

The necessity of reviewing empirical studies on machine learning methods within bug prediction is highlighted by the need for a comprehensive understanding of these techniques. Kamei and Shihab (2016) have contributed to this field by summarizing key aspects of bug prediction, and Wahono (2015) has examined the datasets pertinent to predictive models. Moreover, the review by Jayanthi and Florence (2017) delves into defect prediction techniques, evaluating various software metrics, albeit with certain limitations, including the omission of datasets and performance evaluation methods used. Prasad and Sasikala (2019) also reviewed software defect prediction techniques without detailing the software metrics or performance measures.

My study seeks to cover multiple facets of software bug prediction, with the goal to synthesize, scrutinize, and appraise the machine learning methods employed thus far in the discipline. I will examine the datasets applied to the models, the software metrics typically employed, and the measures used to gauge model performance. By doing so, I aim to identify efficacious techniques and methods applicable for future investigations. The structure of my paper is as follows: Section 2 discusses my methodological approach and defines my research questions; Section 3 presents my findings; Section 4 offers an overview of bug prediction models and enumerates the challenges identified in previous studies; Section 5 acknowledges the limitations of my study; and Section 6 wraps up my discussion with conclusions and avenues for future research.

2. PROCEDURE

The research methodology employed in this study is the Systematic Literature Review (SLR). This method was selected to thoroughly examine the body of research concerning software bug prediction. SLR is a recognized technique that involves the systematic identification, evaluation, and interpretation of research evidence aimed at addressing specific research questions, as outlined by Kitchenham and Charters in 2007.

Research Objective

To structure our review and evaluation of previous studies, we have formulated research questions based on the Population, Intervention, Comparison, Outcomes, and Context (PICOC) criteria as recommended by Kitchenham and Charters in 2007. Table 1 outlines the specific criteria of PICOC.

Table 1: PICOC Criteria

Population	Software, system, application, information system
Intervention	Software bug prediction, software defect prediction, software fault prediction, error-prone, bug-prone, techniques
Comparison	Not available
Outcomes	Positive bug prediction techniques
Context	Small and large datasets, studies in academy and industry

The objective of this review is to assess and synthesize the experimental evidence collected from past studies on the application of machine learning techniques in bug prediction models. The research questions to be addressed in this Systematic Literature Review (SLR) are detailed as follows:

[1]. RQ1 - Which datasets are frequently used for software bug prediction?
[2]. RQ2 - What kind of machine learning techniques that have been selected for prediction model?
[3]. RQ3 - Which metrics are frequently used for software bug prediction?
[4]. RQ4 - Which performance measures are used for software bug prediction?

Review Process:

The methodology for sourcing relevant studies involved selecting digital repositories, crafting a search string, conducting a preliminary search, and extracting a primary list of relevant studies from the repositories that matched the search criteria. The chosen digital repositories for conducting the search were:

[1]. ScienceDirect
[2]. Google Scholar
[3]. SpringerLink
[4]. IEEE Xplore

An exhaustive search method was employed, justified by the relatively small number of primary studies and an even smaller subset focusing on empirical research. The search string—a critical tool for retrieving relevant studies—is crafted by combining various terms and operators. It is designed to maximize the retrieval of pertinent studies. The steps for constructing the search string included:

[1]. Analyzing the research questions with PICOC criteria to identify key search terms.
[2]. Finding significant terms in titles, abstracts, and keywords.
[3]. Considering alternative terminologies for the search terms.
[4]. Applying Boolean operators ("and/or") to structure the search string

The finalized search string was:

Software and (Bug or Fault or Defect) and (Proneness or Prediction) and (Machine Learning or Neural Network or Bayesian Network or Decision Tree or Support Vector Machine or Random Forest)

This search string was utilized across the selected digital databases, with the search confined to the period from 2014 to 2020. This time frame was chosen to focus on the most recent applications of machine learning techniques in research.

To refine the search results, inclusion and exclusion criteria were established as follows:

Inclusion Criteria:

[1]. Studies that analyze software bug prediction models using machine learning.
[2]. Research comparing the performance of different bug prediction models.
[3]. Empirically based studies.
[4]. Studies published in Q1 and Q2 journals.
[5]. Studies written in English.

Exclusion Criteria:

[1]. Studies not related to software bug prediction using machine learning.
[2]. Research lacking a discussion on the performance of bug prediction models.
[3]. Non-empirical studies.
[4]. Studies not published in Q1 and Q2 journals.
[5]. Non-English studies.

From the initial pool of 1,452 studies gathered from the four repositories, titles and abstracts were screened to exclude irrelevant studies, reducing the number to 213. These were further scrutinized using the inclusion and exclusion criteria, ultimately narrowing the field to 31 relevant studies. Table 2 in the document details the distribution of these studies across the different digital repositories.

Table 2: Summary of search results

Repository	Initial list	Second list	Final list
ScienceDirect	226	82	16
Google Scholar	143	22	4
SpringerLink	319	51	8
IEEE Xplore	764	58	3
Total	1452	213	31

Data Extraction

The primary studies sourced from the repositories are essential for addressing the research questions in this SLR. A data extraction method was developed to collect necessary information from these studies to respond to the research questions effectively. Table 3 details the characteristics utilized for answering the research questions, while Table 4 illustrates the alignment between the main studies and the research questions, indicating whether the studies provided answers to the questions.

Table 3: Data extraction characteristics linked to research questions

Characteristic	Research question
Researchers, publications, titles	General
Software bug datasets	RQ1
Software bug prediction machine Learning techniques	RQ2
Software metrics	RQ3
Performance measures for software bug prediction model	RQ4

RESULT

Data Sets

Datasets are collections tailored for specific problems in certain domains. Publicly available datasets like PROMISE and NASA Metrics Data Program are crucial for developing bug prediction models, yet finding standard datasets from organizations is challenging. Pan et al. (2019) addressed public dataset quality issues by creating the Simplified PROMISE Source Code (SPSC) dataset. Researchers use a variety of datasets within different frameworks, making assessments challenging. The PROMISE and NASA datasets are the most employed, featured in 13 studies each. PROMISE provides long-term storage for software engineering data, while NASA MDP offers data on 13 original projects, both aiding predictive model development. Other datasets include AEEM, used in three studies, and relink, used in two, along with datasets from Java, Git, Code4Bench, and Android projects, illustrating the range of data sources in software bug prediction research.

Machine learning Techniques

Many techniques for software bug prediction are presented in the literature and based from the 31 studies, we classified the six most used techniques in software bug prediction.

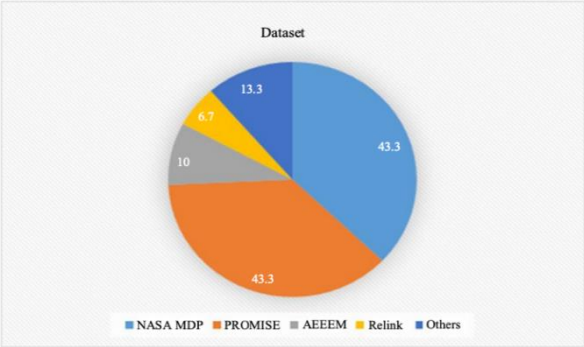

Figure 1: Distribution on Software Bug Datasets

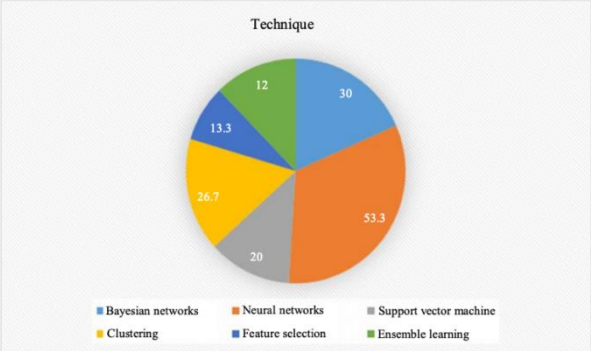

Figure 2: Distribution on machine learning techniques

Table 4: Result of data extraction

Study ID	Reference	RQ1	RQ2	RQ3	RQ4
S1	Erturk and Sezer (2015)	√	√	√	√
S2	Kumar (2018)	√	√	√	√
S3	Pan *et al*. (2019)	√	√	√	√
S4	Zhou *et al*. (2019)	√	√		√
S5	Jin and Jin (2015)	√	√	√	√
S6	Abaei and Selamat (2014)	√	√		√
S7	Okutan and Yildiz (2014)	√	√	√	√
S8	Arar and Ayan (2015)	√	√	√	√
S9	Laradji *et al*. (2015)	√	√	√	√
S10	Rhmann *et al*. (2020)	√	√	√	√
S11	Majd *et al*. (2020)	√	√	√	√
S12	Boucher and Badri (2018)	√	√	√	√
S13	Park and Hong (2014)	√	√	√	√
S14	Jakhar and Rajnish (2018)	√	√	√	√
S15	Ma *et al*. (2014)	√	√	√	√
S16	Ni *et al*. (2017)	√	√	√	√
S17	Kalsoom *et al*. (2018)	√	√	√	√
S18	Miholca *et al*. (2018)	√	√	√	√
S19	Wu *et al*. (2018)	√	√	√	√
S20	Mori and Uchihira (2019)	√	√	√	√
S21	Geng (2018)	√	√	√	√
S22	Dong *et al*. (2018)	√	√		√
S23	Abaei *et al*. (2015)	√	√	√	√
S24	Ryu *et al*. (2015)	√	√	√	√
S25	Rathore and Kumar (2017)	√	√	√	√
S26	Rana *et al*. (2015)	√	√	√	√
S27	Ji *et al*. (2019)	√	√	√	√
S28	Hua *et al*. (2019)	√	√	√	√
S29	Zhao *et al*. (2018)	√	√	√	√
S30	Wei *et al*. (2018)	√	√	√	√
S31	Yang *et al*. (2014)	√	√	√	√

Figure 2 illustrates the methods and distribution of studies on bug prediction modeling techniques. Although numerous studies evaluate these techniques, no consensus exists on the optimal method when viewed individually. The primary techniques include Bayesian Network (BN), Neural Network (NN), Support Vector Machine (SVM), Clustering, Feature Selection (FS), and Ensemble Learning (EL), with Neural Networks being most prevalent. Arar and Ayan (2015) highlighted challenges with NN due to parameter selection difficulties, suggesting a combination of ANN with the Artificial Bee Colony (ABC) algorithm to optimize parameters. Miholca et al. (2018) also developed a framework that merges ANN with gradual relational association rules for identifying defective software entities.

Software Metrics

Software metrics serve as crucial independent variables for predicting bug proneness, as showcased in Figures 3 and 4, which detail the type and usage frequency of these metrics in primary studies. Key metrics include McCabe metrics, introduced in 1976, which measure complexities like Cyclomatic, Essential, and Design Complexity. Line Of Code (LOC) metrics, evaluating aspects such as lines, comments, and their combinations, are used in half of the studies and are most effective when combined with other metrics. Other widely used metrics are Halstead metrics, CK Metrics Suite for object-oriented characteristics, QMOOD metrics for assessing design properties, and Martin's metrics for evaluating design quality.

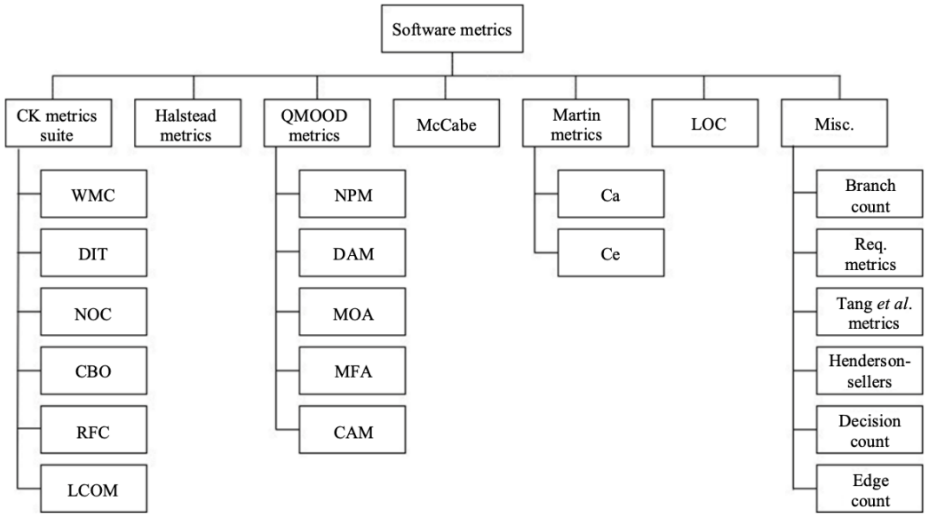

Figure 3: Type of metrics

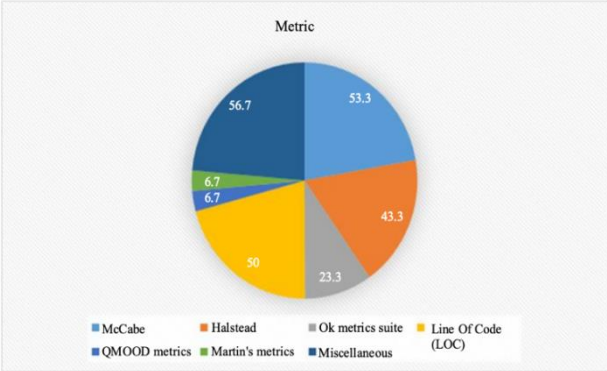

Figure 4: Distribution of software metrics

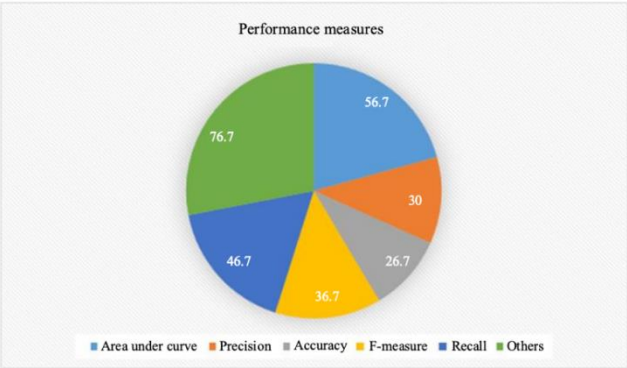

Figure 5: Distribution of performance measures

VALIDITY

The purpose of this study is to analyze the past studies on software bug prediction using the machine learning techniques. Most of the studies have a huge range of datasets, but I cannot be sure whether these datasets represent the bug prediction scenarios or not. For this study, I did not resort to manual reading of titles of all published papers in journal during the searching stage. In fact, I used the search string that I had constructed earlier to find the relevant studies on bug prediction. I have search as many studies as I could in accordance to inclusion and exclusion criteria. However, there is a likelihood that I had overlooked other proper studies. Also, this review did not include the studies from conference proceedings since I only focused on papers from the primary journals. Therefore, it had limited other machine learning techniques for my review. The final concern is about the researcher bias, where they have the tendency to confirm that the written information was true.

CONCLUSION

In this study, I conducted a review so that Icould analyze and evaluate the performance of software bug prediction model using machine learning techniques. After a detailed investigation followed by an orderly step, I identified 31 main studies within the period of 2014 to 2020. I summarized the studies based on the datasets, machine learning techniques, software metrics and performance evaluation measurements. The main findings that I have gotten from the main studies are summarized as below: [1]. NASA MDP and PROMISE repositories were the most frequently used dataset in the past literature [2]. BN, EL, FS, NN, Clustering and SVM were the machine

learning techniques that I have identified and the most widely used technique for bug prediction model were NN and BN [3]. CK Metrics Suite was found to be the most widely chosen as independent variables in the past literature. CBO, RFC and LOC were found to be the most useful metrics in bug prediction domain [4]. AUC, precision, recall, F-Measure and accuracy are the most frequently used performance measures in the main studies The following are the recommendations for future research on software bug prediction using machine learning techniques: [5]. There are a few studies that adopt the software bug prediction for agile development [6]. There are a few studies that improve the performance of bug prediction models through integration with other algorithms [7]. There are few studies that proposed an approach to make the models more informative

REFERENCES

[1]. Andresen, B. H., Casasanta, J. A., Keeney, S. C., Martin, R. C., & Satoh, Y. (1994). U.S. Patent No. 5,355,037. Washington, DC: U.S. Patent and Trademark Office.
[2]. Abaei, G., &Selamat, A. (2014). Increasing the accuracy of software fault prediction using majority ranking fuzzy clustering. International Journal of Software Innovation (IJSI), 2(4), 60-71.
[3]. Abaei, G., Selamat, A., & Fujita, H. (2015). An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction. Knowledge-Based Systems, 74, 28-39.
[4]. Al-Jamimi, H. A. (2016, August). Toward comprehensible software defect prediction models using fuzzy logic. In 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS) (pp. 127-130). IEEE.
[5]. Arar, Ö. F., & Ayan, K. (2015). Software defect prediction using cost-sensitive neural network. Applied Soft Computing, 33, 263-277.
[6]. Bansiya, J. & Davis, C. (2002). A hierarchical model for object-oriented design quality assessment. IEEE Transactions on Software Engineering, 28(1), 4-17. Boucher,
[7]. A., & Badri, M. (2018). Software metrics thresholds calculation techniques to predict faultproneness: An empirical comparison. Information and Software Technology, 96, 38-67.
[8]. Chidamber, S. &Kemerer, C. (1994). A metrics suite for object-oriented design. IEEE Transactions of Software Engineering, 20(6), 476-493.
[9]. Couto, C., Pires, P., Valente, M. T., Bigonha, R. S. & Anquetil, N. (2014). Predicting software defects with causality tests. Journal of Systems and Software, 93, 24-41.
[10]. D'Ambros, M., Lanza, M., &Robbes, R. (2010, May). An extensive comparison of bug prediction approaches. In 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010) (pp. 31-41). IEEE
[11]. Dong, F., Wang, J., Li, Q., Xu, G., & Zhang, S. (2018). Defect prediction in android binary executables using deep neural network. Wireless Personal Communications, 102(3), 2261-2285.
[12]. Erturk, E., &Sezer, E. A. (2015). A comparison of some soft computing methods for software fault prediction. Expert systems with applications, 42(4), 1872-1879.
[13]. Geng, W. (2018). Cognitive Deep Neural Networks prediction method for software fault tendency module based on Bound Particle Swarm Optimization. Cognitive Systems Research, 52, 12-20.
[14]. Halstead, M. H. (1977). Elements of software science (Vol. 7, p. 127). New York: Elsevier. Hassan, F., Farhan, S., Fahiem, M. A., & Tauseef, H. (2018). A Review on Machine Learning Techniques for Software Defect Prediction. Technical Journal, 23(02), 63-71.
[15]. Hua, W. E. I., Chun, S. H. A. N., Changzhen, H. U., ZHANG, Y., & Xiao, Y. U. (2019). Software Defect Prediction via Deep Belief Network. Chinese Journal of Electronics, 28(5), 925-932.
[16]. Jacob, S. G., & Raju, G. (2017). Software defect prediction in large space systems through hybrid feature selection and classification. Int. Arab J. Inf. Technol., 14(2), 208-214.
[17]. Jakhar, A. K., & Rajnish, K. (2018). Software fault prediction with data mining techniques by using feature selection based models. International Journal on Electrical Engineering and Informatics, 10(3), 447-465.
[18]. Jayanthi, R. F., & Florence, L. (2017). A review on software defect prediction techniques using product metrics. International Journal of Database Theory and Application, 10(1), 163-174.
[19]. Ji, H., Huang, S., Wu, Y., Hui, Z., & Zheng, C. (2019). A new weighted naive Bayes method based on information diffusion for software defect prediction. Software Quality Journal, 27(3), 923-968.
[20]. Jin, C., &Jin, S. W. (2015). Prediction approach of software fault-proneness based on hybrid artificial neural network and quantum particle swarm optimization. Applied Soft Computing, 35, 717-725.
[21]. Kalsoom, A., Maqsood, M., Ghazanfar, M. A., Aadil, F., & Rho, S. (2018). A dimensionality reduction-based efficient software fault prediction using Fisher linear discriminant analysis (FLDA). The Journal of Supercomputing, 74(9), 4568-4602.
[22]. Kamei, Y., & Shihab, E. (2016, March). Defect prediction: Accomplishments and future challenges. In 2016 IEEE 23rd international conference on software analysis, evolution and reengineering (SANER) (Vol. 5, pp. 33-45). IEEE.

[23]. Kaur, G. & Sharma, D. (2015). A study on Robert C. Martin's metrics for packet categorization using fuzzy logic. International Journal of Hybrid Information Technology, 8(12), 215-224.
[24]. Kazienko, P., Lughofer, E., &Trawiński, B. (2013). Hybrid and ensemble methods in machine learning J. UCS special issue. J UniversComput Sci, 19(4), 457-461.