IJCSC 0973-7391 Volume 8 • Issue 1 Sept 2016 - March 2017 pp. 67-75 available online at www.csjournals.com

Machine Learning-Based Software Bug Prediction: A Review

Er. Seema Rani
Assistant Professor, Computer Science and Engineering, CDLSIET, Sirsa

Abstract – Software is used to operate a wide range of necessary systems and devices in today's world. Consequently, many companies create systems with different sizes and functions in an attempt to provide higher-quality software more quickly. The aim of this diversity in software development is to effectively meet the client's goals while adjusting to the many demands of the current technological environment. Early software bug prediction improves software's quality, dependability, efficiency, and cost. Because software development and maintenance operations are concerned with the overall success of software, software bug prediction (SBP) is a crucial subject. Any software product or program that serves a commercial industry like production, aeronautics, medical, financial coverage etc. is referred to as software system. Software quality is determined by how well the program adheres to its design and how well it is constructed. When assessing software quality, some of the factors we consider are accuracy, quality, expendability, completeness, and lack of errors. Since different organizations use different quality standards, it is preferable to use program metrics which are a better way to assess program quality. Software metrics-collected attributes from source code can be used as an input for software defect predictors. Errors introduced by stakeholders and software developers are known as software defects. Through a careful analysis of the collection of existing research, the goal is to investigate current trends in software bug prediction. The review concludes by highlighting broad range of machine learning applications on software bug that we discovered during earlier study projects.

Keywords: Software Defect, Software Engineering, Bug Prediction, ML Techniques.

1. INTRODUCTION

The presence of flaws in software has a significant impact on its dependability, quality, and upkeep expenses. Even with correctly applied software, it takes labor to achieve bug-free software because hidden defects are common. A significant issue in software engineering is also creating software bug prediction models that could identify problematic modules early on. A bug is any malfunction, mistake, or breakdown in software. It exhibits undesirable behavior and either yields an incorrect or surprising outcome. It is an unexpected behavior caused by a flaw in a software product [1]. A software flaw inevitably results from a system's software failing on a regular basis over time. Errors introduced by stakeholders and software developers are known as software defects. Improving software product quality while reducing their cost and time is the primary goal of software defect prediction. A software defect, often known as a bug, is a flaw in the software product that prevents it from carrying out the function that the developer and user intended. One of the most important and inspiring areas of study is machine learning, which aims to extract valuable information from massive data sets. Finding patterns in data that may be used is the fundamental goal of machine learning. For example, structured data mining can produce structured data & unstructured data. Here, we closely examined primary causes of

failures that result in defects, cost of the software, time needed for testing and maintenance after delivery to stakeholders. Furthermore, we look at machine learning principles, suggested fixes for software errors and software engineering applications of machine learning particularly for software testing and maintenance. Section 1 presents the research paper. Section 2 delineates major factors contributing to software failures and presents the researcher's recommended filter**.** Commonly known defect predictors were compiled in Section 3. Machine learning ideas and their applications were covered in Section 4. Section 5 Finally, we attempted to evaluate the machine learning research works concerning software bugs and categorized them according to the techniques they employed, specifically the classification method, clustering method, and ensemble approaches. In a summary, the researchers offer recommendations for future study directions. Section 6 concludes the research.

2. KEY SOFTWARE FAILURE FACTOR

A software system is any software product or program that supports a commercial domain, such as social networking, e-commerce, finance, healthcare, insurance, manufacturing, or any other domain. Software system development and design requires funding, domain-specific experts, a significant amount of time, tools, and infrastructure. Even though software companies have extensive expertise in designing and implementing projects, but frequency of software failures is on the rise, which results in lost time, money, and energy. The system can malfunction as a result of a bug that occurs during each SDLC, or the client might not deliver precise specifications since unfamiliar with information technology initiatives.

In addition, the survey respondents were questioned about the elements that lead to project challenges. We have found that the two main things that make a project effective are inadequate customer requirements and a lack of user input. The following diagram shows the main causes of software failure:

- Insufficient User Input[2]
- Uncertain Aim[3]
- The Incomplete demands and conditions[4]
- Varying Necessities & Conditions
- Insufficient administrative support[5]
- Technology Incompetence
- Unrealistic Expectations[6]
- Impractical Time Frames
- Latest Technology

3. SOFTWARE DEFECT PREDICTOR

One approach or technique that supports software development life cycles and software testing is the software defect predictor [7]. A software defect, often known as a bug, is a flaw in the software product that prevents it from carrying out the function that the developer and user intended. IEEE standard lists several categories for software errors [8], including:

- **Defect:**
  It happens when an application doesn't work as required. Deviation between the application/ software's actual and intended results is called default. Tasks and

requirements provided by developer and customers are not being performed. In other words, we can say that the bug declared by the programmer and inside the code is called a Defect.

- **Bug:**
  In software testing a bug is casual word of defect, signifies that s/w is not performing as per the prerequisites. Coding error makes a program to malfunction and referred as a bug. Term "bug" is used by the test engineers. A QA (Quality Analyst) can use the bug report template to help them recreate and document any bugs they find.
- **Error:**
  The customer may be experiencing this because they are aware of the software's shortcomings, which lead to inaccurate outcomes. Code problems can result in errors, meaning that a developer's coding fault may have happened because the developer misinterpreted the need or failed to specify it correctly. Developers refer to this as an error.
- **Failure:**
  Software may have a bug if fault tolerance coding has not been added, which could cause an application to malfunction. A program may experience a bug for the reasons like insufficient resources, improper action and unsuitable definition of the data.
- **Fault:**
  Numerous flaws cause software to malfunction, meaning that a loss indicates a serious problem with the program, application, or one of its modules, rendering the system unusable or malfunctioning. Put differently, we can argue that a specific problem with a product is deemed a failure if it is discovered by an end-user. It is possible that a single flaw could result in one or more failures.

4. BUG PREDICTION USING MACHINE LEARNING

One of the primary distinctions between human and computer labor is that human workers typically expand efforts to enhance their performance while engaging in any given task. This indicates that, unlike machines, humans are capable of performing any work with adaptability. With the help of prior instances of correlations between input data and outputs, machine learning algorithms "learn" to anticipate outputs. By testing, evaluating, and making corrections when necessary, the models built based on the link between inputs and output gradually improved [9]. In accordance with Tom Mitchell's description, machine learns accoding to specific tasks T, performance P, and Experience E [10]. One use of artificial intelligence (AI) is in machine learning. Thanks to advances in machine learning, computer systems can now be trained on historical data to acquire new skills and grow more adept at doing specific tasks. The term "cognitive system" refers to a system that uses a model to simplify the environment and help grasp concepts and their surroundings [11]. We call this process of building the model "inductive learning." By creating new patterns and structures, the cognitive system can integrate its experiences. Machine learning is the process by which a cognitive system constructs a model or pattern. By creating new patterns and structures, the cognitive system can integrate its experiences. While informative patterns are characterized by just describing a fraction of the data and predictive models are defined as those that can be applied to predict, the output of a function (target function). Semi-supervised, supervised, unsupervised, and reinforcement

learning are the types of machine learning [12]. We looked at supervised and unsupervised learning below in sections a, b.

(a)Supervised Learning

A machine learning task is to derive a function from labeled training data. The training data is an assortment of training examples. Every example is made up of two parts: an input item and intended output. The identified tasks for supervised learning are classification and regression. While classification deals with the creation of prediction models for functions with discrete ranges, regression deals with the construction of continuous range models. Many machine learning researchers are interested in supervised learning. Among popular supervised ML approaches include concept learning, instance-based learning, Bayesian learning, classification, linear regression, neural networks, and SVM.

(b)Unsupervised Learning

This is often referred to as observational learning. Unsupervised learning requires the system to find patterns only by using the shared attributes of the sample without knowing how many patterns exist. Popular methods here are clustering, sequential pattern mining, and association rule mining.

Machine learning is not a challenging scientific field [13]. Software engineers can employ machines to reduce the time and expense of the system development phase since they can learn automatically from training data and build smaller versions of current systems or data summaries. Creating machine learning-based solutions for software engineering issues is one way to get around the integration of machine learning and software engineering. Similar to other applications, software engineering requires pre-processing of the data and pattern complexity before implementing machine learning techniques. Developers are currently paying close attention to component-based approach fault prevention methods, which break down projects into individual components to improve quality and reliability while reducing development time and cost [14]. However, these methods are only useful for identifying issues with individual component quality. We used factors including software development effort, software dependability, and programmer productivity to quantify software quality and forecast the significance of models in software engineering. Research was done on early software quality prediction to improve system performance using machine learning approaches (fuzzy logic and case-based reasoning) [15]. Research indicates that the following software engineering issues are amenable to machine learning resolution: project management, software testing, requirements gathering, software reuse qualification, software measurement selection, defect prediction models, and software testing. We also referred to it as a machine learning application in software engineering [16]. Of all the software engineering topics we identified, the researcher chose to focus on software defect prediction using machine learning approaches for this study. Machine learning has been expanding quickly, leading to the development of numerous learning algorithms for various uses. The degree to which those algorithms are successful in resolving

real-world issues determines their eventual worth. Thus, replication of algorithms and their application to novel tasks are essential for the field's advancement [17]. Currently, meanwhile, a number of machine learning researchers publish for the creation of software defect prediction models. We now divide the successful software defect model into three categories: ensemble, clustering, and classification approaches.

5. RELATED WORK

According to Ezgi Erturk et al., The PROMISE Software Engineering Repository provided the data set for the experiment, and McCabe software metrics were used [18]. SVM, ANN, and ANFIS (a newly introduced adaptive model) are the algorithms they used in the experiment; the corresponding performance measures were 0.7795, 0.8685, and 0.8573.

Malkit Singh et al., Early software testing methods for investigating software faults included building a model using a neural network tool based on the Levenberg-Marquardt (LM) algorithm using data from the PROMISE and then contrasting LM accuracy with that of a neural network based on polynomial functions. Levenberg-Marquardt (LM) had a better accuracy (88.1%), according to the testing. Therefore, the machine learning based on neural networks has good accuracy.

In this study, Saiqa Aleem et al.,employed a variety of machine learning techniques on about 15 different data sets like AR1, CM1, KC1, etc. Evaluated each method's effectiveness and came to the conclusion that SVM, MLP, and bagging given the best accuracy results [19].

Martin Shepperd et al. [20] conducted an analysis and used a novel benchmark system to anticipate and evaluate software defects. Various learning systems are assessed in the evaluation step based on the chosen scheme. Next comes the prediction stage, when all past data is utilized to create a predictor using the best learning scheme, which is then used to predict defects in the incoming data.

In order to enhance the model's performance, Xi Tan et al. [21] experiment with a function cluster-based software defect prediction model. The researcher increases recall value by 99.2% & precision by 91.6%.

Jaspreet Kaur and associates [22] used a k-mean based clustering technique to investigate how error-prone object-oriented programming is. Their findings led them to draw a conclusion, as they achieved 62.4% accuracy.

Shanthini et al. built models utilizing an ensemble technique, with the aim of addressing software failure prediction through the use of an ensemble approach. Three categories were used to categorize the data set: method, class, and package levels. For both method and class level measures, they used NASA KC1 data, and for package level metrics, they used eclipse data with ensemble methods. The research's finding shows that bagging executes superior for data at the procedure and package levels. The technique level findings are 0.809 for bagging, 0.782 for boosting, 0.79 for staking and 0.63 for voting using AUC-curve measurement.

Similarly, for package level data, the AUC-curve performance measures are 0.82 for bagging, 0.78 for boosting, 0.72 for staking, and 0.76 for voting [23].

YI PENG et al., Purpose was to use an analytical hierarchal procedure to evaluate quality of collective techniques in SFP. They used 10 publicly available NASA data sets and 13 performance indicators. In this paper, Bagging, Boosting, and Staking were an ensemble method. Decision trees are base classifiers in this instance since their performance metric, Ada Boost, yields the best result accuracy of 92.53%.To enhance clarity, we have compiled the aforementioned works into Table IV, which includes the goal, the methodology used, the studies' contribution, and an overall commentary on the studies.

Table I: Summary of Related Works

Author (Year)	Objectives	Methodology/Approach/ Tools/Techniques	Key Findings	Remarks
Ezgi Erturk et al (2014)	To find s/w bugs.	-SVM, ANN, ANFS. -Cross validation test Model. -WEKA tool.	-Performance of the model is measured. -SVM 0.7795, -ANN 0.8685 -ANFS 0.857.	NASA data set is used and maximum accuracy they scored is 0.857.
Malkit Singh et al (2013)	Early in SDLC, software problems are predicted.	-Levenberg-Marquardt (LM) Algorithm - ANN. -Polynomial Function -MATLAB	-Accuracy with LM based is 88.1 % -Accuracy with PF is 78.8 %.	ANT-1.7data set is used and accuracy achieved is 88.7%.
Saiqa Aleem et al.(2015)	Comparative ML techniques employing software prediction models for publically data.	-10 Cross validation test -SVM -Ada Boost -Bagging -Random Forest	-Accuracy for SFP model using -SVM 89.29 % -Bagging 89.38% -Random forest 89.08 %	NASA data set is used. Accuracy they scored is 99.52 for PC2data.
Martin Shepperdet al.(2014)	To predict the factors influencing the accuracy of SFP.	Meta-analysis is done for the previous studies on SFP which are relevant and good quality studies.	They come to the conclusion that studies on defect prediction ought to focus on blind analysis and enhance intergroup and reporting processes.	It was a survey on the factors that affect performance of SFP accuracy.
Xi Tan etal.(2011)	To recover the software fault prediction model's performance through the usage of recall and accuracy.	-Eclipse 3.0 data -90 to 10 % data split	in terms of recall and precision, cluster-based defect prediction outperforms over class-based models. -Recall (31.6% to 99.2%) -Precision (73.8%	Eclipse3.0 data is used and scored the accuracy 99.2 %.

			to91.6%	
QinbaoSong et al.(2011)	A general methodology for assessing software failure prediction models was devised and evaluated by them.	-NASA datasets -2 data preprocessors -2 attribute selections -NB	The highest accuracy achieved with the Naïve Bayesian technique on the PC1 data set was 89.7%.	For PC1 data, the maximum accuracy using NB is 89.7.
Jaspreet Kaur et al.(2011)	They utilized K-mean to predict how prone to errors object-oriented programming is.	-KC1 data set -WEKA tool -K-mean algorithm	Scored accuracy is 62.4 %.	Through KC1data set scored maximum accuracy is 62.4.
Mikyeong Park et al.(2014)	To anticipate s/w errors.	-3 promise repository -X-Means	X means have higher accuracy (90.48) for AR3.	Author scored accuracy 90.48.
Shanthini. A et al.(2013)	Software defect prediction with a collective method.	-NASA KC1 Data -WEKA Tool -Bagging -Staking -Voting	-Bagging perform superior when compared with other. -Bagging (0.809) -Staking (0.79) -Voting (0.63)	Class level data is used & max accuracy is AUC-0.809.

6. CONCLUSION AND FUTURE WORK

Software system development is emerging more and more these days compared to earlier years. But before the product is given to end users, quality control must be done. To enhance the software quality, quality metrics such as CMM, ISO standards, and software testing are used. These days, testing plays an increasingly significant role in program reliability. Predicting software flaws can help software testing run more smoothly and help with resource allocation. We ought to devote more time and provides resources to the error-prone modules. Research's primary goal was to evaluate earlier studies on software defects that apply machine learning techniques, data sets, tools, methodology, contributions to science, and classification systems.

REFERENCES

[1] Seema Rani," A Comparative Study Of Automation Tools And Frameworks Automated Software Testing" in International Journal of Computer Science and Communication(IJCSC),Volume 7, Issue-2, Mar-Sept 2016.

[2] Venkata U and R. A, "Empirical Assessment of Machine Learning based Software Defect Prediction Techniques", Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, 2005.

[3] Robert N, Akmel ,"Why Software Fails ", International Journal of Emerging Research in Management &Technology,Volume-6, Issue-6, 2005. ISSN: 2278-9359.

[4] Malhotra, Ruchika, and Yogesh Singh, "On the applicability of machine learning techniques for object oriented software fault prediction",Software Engineering: An International Journal vol 1.1, Pp 24-37, 2011.
[5] L. J, "Major Causes of Software Project Failures ", CROSSTALK The Journal of Defense Software Engineering, pp. 9-12, July 1998.
[6] D'Ambros, Marco, Michele Lanza, and Romain Robbes, "An extensive comparison of bug prediction approaches." Mining Software Repositories (MSR), 7th IEEE Working Conference, 2010.
[7] Vikas S and J. R, "Cataloguing Most Severe Causes that lead Software Projects to Fail", International Journal on Recent and Innovation Trends in Computing and Communication, vol. 2, Pp 1143–1147, May 2014.
[8] Mikyeong P and E. H, "Software Fault Prediction Model using Clustering Algorithms Determining the Numberof Clusters Automatically", International Journal of Software Engineering and Its Applications, vol. 8, pp. 199-204, 2014.
[9] R. Malhotra, "Comparative analysis of statistical and machine learning methods for predicting faulty modules," Applied Soft Computing 21, (2014): 286-297
[10] Malhotra, Ruchika. "A systematic review of machine learning techniques for software fault prediction", Applied Soft Computing 27, Pp 504-518, 2015.
[11] George T, Ioannis K, Ioannis P, and Ioannis V, "Modern Applications of Machine Learning ", Proceedings of the1st Annual SEERC Doctoral Student Conference vol. 1,2006.
[12] Sarwesh S and S. K, "A Review of Ensemble Technique for Improving Majority Voting for Classifier", International Journal of Advanced Research in Computer Science and Software Engineering,vol. 1, Pp. 177-180, January 2013.
[13] T. M. Mitchell, “The Discipline of Machine Learning”, Carnegie Mellon University, 2006.
[14] Xia C, Michael R, Kam-Fai W, and M. W, "Compare: A Generic Quality Assessment Environment for Component-Based Software Systems," Center of Innovation and Technology the Chinese University of Hong Kong, pp. 1-25.
[15] Ekbal R, Srikanta P, and V. B, "A Survey in the Area of Machine Learning and Its Application for Software Quality Prediction, "ACM SIGSOFT Software Engineering, vol. 37, September 2012.
[16] 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.
[17] Yogesh S, Pradeep K, and O. S, "A Review of Studies on Machine Learning Techniques," International Journal of Computer Science and Security, vol. 1, pp. 70-84.
[18] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," Expert Systems with Applications, 2014.
[19] Saiqa A, Luiz F, and F. A, "Benchmarking Machine Learning Techniques for Software Defect Detection", International Journal of Software Engineering & Applications, vol. 6, pp. 11-23, May 2015.

[20] Martin S, David B, and T. H, "Researcher Bias: The Use of Machine Learning in Software Defect Prediction", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, vol. 40, pp. 603-616, JUNE 2014.
[21] X. Tan, X. Peng, S. Pan, and W. Zhao, "Assessing Software Quality by Program Clustering and Defect Prediction," pp. 244-248, 2011.
[22] Jaspreet K and P. S, "A k-means Based Approach for Prediction of Level of Severity of Faults in Software System," Proceedings of International conference on Intelligent Computational Systems, 2011.
[23] Pooja P and D. A. Phalke, "Software Defect Prediction for Quality Improvement Using Hybrid Approach", International Journal of Application or Innovation in Engineering & Management, vol. 4, June 2015.