ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

An Extensive Review of Machine Learning Approaches in Predicting Software Defects: Insights from the Literature

Dr E. SriDevi[1] ,

Asst. Professor, Department of C.S.E, Koneru Lakshmaiah Education Foundation, Guntur, A.P, India – 522502.

V.PremaLatha[2]

Assoc. Professor, Department of C.S.E, Koneru Lakshmaiah Education Foundation, Guntur, A.P, India – 522502.

Abstract:

Any software program or set of applications that assist commercial domains such Aviation, manufacturing, health care, insurance, and so forth. Software quality is determined by how well the program adheres to its design and how well it is constructed. Several of the factors that we are Correctness, Product quality, Scalability, Completeness, and Bug-Free are the criteria for evaluating software quality. But because every organization has a different set of quality standards, it is preferable to Utilize software metrics to gauge the product's quality. Software defect predictors can use attributes that we collected from source code using software metrics as an input. Errors introduced by stakeholders and software developers are known as software defects. Ultimately, this study revealed how machine learning may be applied to software defects, a finding gleaned from earlier research projects.

Introduction

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 costs and time is the primary goal of defect prediction. A software defect, often known as a bug, is a flaw in the software product that prevents it from doing the function that the developer and user intended. One of the most important and inspiring fields of study is machine learning, which aims to extract valuable information from massive data sets. Machine learning's primary goal is to extract meaningful patterns from data. Text mining is the process of extracting usable patterns from unstructured data, such as natural language documents, or structured data, such as various

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

databases. In this study, we closely examined the primary causes of software failures that result in software defects, costs incurred by the Software Company, and turnaround times for testing and maintenance after delivery to stakeholders. We also look at the suggested fixes for software bugs, the principles of machine learning, and how machine learning is used to software engineering—more especially, software testing and maintenance. The remainder of this essay is structured as follows: Section 4 talked about machine learning concepts and the application area of machine learning with regard to software engineering, specifically for software defects. Section 2 explained the main causes of software failures and the recommendations filtered out by the researcher. Section 3 summarized common known defect predictors. In the end, we made an effort to evaluate the machine learning research papers about software defects and categorized them according to the techniques they employed—that is, using ensemble, clustering, and classification approaches. The researchers offer a summary and recommendations for further research in their conclusion to this paper.

Principal causes of software malfunctions

Any software product or program that supports a business domain, including manufacturing, banking, insurance, healthcare, aviation, social networking, e-commerce, or any other domain, is referred to as a software system [1]. Software system development and design requires funding, domain-specific experts, a significant amount of time, tools, and infrastructure. Even though software companies have vast expertise designing and implementing projects. A problem that occurs during each software development cycle could lead the system to fail, or the customer might not provide you the precise requirements because they are unaware of political or cultural challenges or information technology projects. In addition, the survey respondents were questioned about the elements that lead to project challenges. Lack of user involvement[1][2][3], unclear goals and objectives[3][4], incomplete requirement specification[3][4], lack of resources[3], inadequate project planning and scheduling [3-5], poor team member communication[3][4], and poor testing are the most frequently cited causes of software failures. As can be seen from Table 1 below, the two main things that contribute to a project's success are inadequate user requirements and a lack of user feedback.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

2.1 Suggestion for Handling Software Errors

Although the causes of software failures were covered in Section II, there is always a solution to a problem; therefore here are some suggestions for making projects successful.

- Keep an eye on the project and the estimate's accuracy.
- Completely met the needs of the user.
- Never rely solely on a timetable or expense estimate.
- State the goal and objectives clearly.

Predictor for Software Defects

One approach or technique that supports software development life cycles and software testing is the software defect predictor. Software defects, often known as bugs, are defects in software products that prevent the software from doing its intended function as intended by the developer and the user. The Institute of Electrical and Electronics Engineers (IEEE) standard 104[8] lists several categories for software errors. These categories are:

- Defect: The inability to complete the tasks and meet the system criteria that consumers and developers have specified.
- Error: This may occur from the customer's awareness of the software's shortcomings, leading to inaccurate outcomes.
- Failure: The software products will no longer perform as necessary, or inaccurate results will be displayed for each consumer input.
- Fault: the software goods have a glaring or evident fault.

According to the aforementioned categories for software defects, a flaw in a software product might cause it to malfunction or cause health issues in the case of significant and important software, like NASA software products for space sciences.

1. Machine learning concepts

One of the primary distinctions between human and computer labor is that, when undertaking any type of task, humans typically broaden their efforts to enhance their performance at the same time. This suggests that, unlike machines, humans are versatile enough to execute any work. With the help of prior instances of correlations between input data and outputs, machine learning algorithms may "learn" to anticipate outputs. By testing

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

the models and making corrections when necessary, the ones built on the basis of the relationship between input and output gradually improved[9]. In accordance with Tom Mitchell's description, the machine learns in relation to specific tasks T, performance P, and experience E[10].

A cognitive system is one that makes use of a model to simplify the environment in order to comprehend a notion and its 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 and pattern. While informative patterns are characterized by just describing a piece of the data, predictive models are defined as those that may be used to forecast the output of a function (target function) for a given value in the function's domain. Four categories exist for machine learning tasks: reinforcement learning, unsupervised learning, semi-supervised learning, and supervised and unsupervised learning. Of them, supervised and unsupervised learning is the most well-known [8][12][12]. We looked at some supervised and unsupervised learning below in sections a and b.

A) Supervised learning

Deriving a function from labeled training data is a machine learning task. A collection of training examples makes up the training data. Each example in supervised learning consists of a pair: an input item and a desired output value. Its two recognized supervised learning tasks are regression and classification. Regression is concerned with creating continuous range models, whereas classification is concerned with creating predictive models for functions with discrete ranges. Supervised learning is the area of interest for many machine learning researchers [13]. Concept learning, classification, rule learning, instance-based learning, Bayesian learning, liner regression, neural networks, and support vector machines are among the most popular supervised machine learning techniques.[8,11, 14].

B) Unsupervised learning

This is often referred to as observational learning. Without knowing how many or even whether there are any patterns at all, the system in unsupervised learning must investigate any patterns based simply on the shared characteristics of the example. Clustering, sequential pattern mining and association rule mining are the most often used techniques in unsupervised learning.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

Machine learning is not a challenging scientific field [15]. 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 [11]. Similar to other applications, software engineering requires pre-processing of the data and pattern complexity before implementing machine learning techniques. Developers are paying a lot of attention these days to component-based approach fault prevention techniques, which break down large projects into smaller, more manageable components and save time and money in the process. Nevertheless, these techniques are only useful for identifying issues with individual component quality [16]. Software engineering using a machine learning technique is one way to overcome the issue. We used factors including software development effort, software reliability, 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) [17]. Research indicates that the following software engineering issues are amenable to machine learning resolution: project management, software testing, software measurement selection, defect prediction models, software reuse qualification, requirements gathering, and software quality estimation.In [9][18][19][20][21], we also referred to it as a machine learning application in software engineering. The researcher decided to carry out this study on software defect prediction using machine learning techniques out of all the software engineering difficulties we specified.

Machine learning is a rapidly expanding discipline that has produced a wide range of learning algorithms for various uses. Software engineering is one area where machine learning is used, as we saw in section 2.5.4.1. The degree to which those algorithms are successful in resolving real-world issues determines their eventual worth. Consequently, replication of algorithms and their application to novel tasks are essential for the field's advancement. Nonetheless, a number of machine learning researchers are actively publishing for the creation of software defect prediction models. We have now divided the successful software defect model into three categories: ensemble, clustering, and classification approaches.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

2. Based on Classification Methods

The data set for the experiment was gathered from the PROMISE Software Engineering Repository and McCabe software metrics were applied, according to EzgiErturk et al. [22]. SVM, ANN, and ANFIS (a new adaptive model presented) are the algorithms they used in the experiment; the corresponding performance measures were 0.7795, 0.8685, and 0.8573.

Surndha Naidu et al.[23] released another successful paper, the main objective of which was to determine the overall number of flaws in order to reduce time and expense. Five criteria, including volume, program length, difficulty, effort, and time estimator, were used to categories the defect. They classified flaws using the ID3 classification technique.

Singh Malkit et al. [52] 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 repository of empirical software engineering data, and then contrasting the accuracy of LM with that of a neural network based on polynomial functions. Levenberg-Marquardt (LM) had a better accuracy (88.1%), according to the testing. Thus, the machine learning based on neural networks has good accuracy.

According to Saiqa Aleem et al. [24], around fifteen data sets (AR1, AR6, CM1, KC1, KC3, etc.) were employed in this work along with a variety of machine learning techniques. After evaluating each method's performance, it was determined that SVM, MLP, and bagging offered the best accuracy and performance.

Venkata U.B et al.[2] state that they assess various predictive models for data sets containing real-time software defects. The experiment demonstrated that no single methodology is perfect for every collection of data, but IBL and 1 R consistently produced predictions with higher accuracy than other approaches.

Martin Shepperd et al.'s [25] analysis states that they used a unique benchmark framework to anticipate and evaluate software defects. Various learning systems are assessed in the evaluation step based on the chosen scheme. Next, using all of the previous data, the best learning scheme is used to create a predictor, which is then used to predict defects in the new data.

3. Based on Clustering Methods

In order to enhance the model's performance, Xi Tan et al.[26] experiment with a function cluster-based software defect prediction model. The researcher improves performance with this strategy from 31.6% to 99.2% recall and from 73.8% to 91.6%

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

precision. Jaspreet Kaur et al.'s investigation [27] into the fault proneness of object-oriented programming using a clustering approach based on k-mean analysis led them to the conclusion that their accuracy rate was 62.4%.To forecast software errors, a model is constructed utilizing three promise repository datasets (AR3, AR4, and AR5) and clustering algorithms (EM and X-means). The data set was first normalized to range from 0 to 1, and then the CfsSubsetEval attribute selection procedure was used without attribute reduction. The experiment's findings demonstrated that, for AR3 without attribute reduction, X-means had higher accuracy (90.48) than other models [27].

4. Based on Ensemble Approach

Shanthini et al. conducted model building using an ensemble technique. The aim of this research was to address software failure prediction through the use of an ensemble approach. Three levels were used to categories the data set: method, class, and package levels. For method and class level measures, they were using NASA KC1 data, and for package level metrics, they were using eclipse data with ensemble methods (bagging, boosting, staking, and voting). The outcome of the experiment demonstrates that bagging works better for data at the method and package levels. The AUC-curve performance measuring method level results for bagging (0.809), boosting (0.782), staking (0.79), and voting (0.63) were as follows. The package level data's AUC-Curve performance measures were similarly bagging (0.82), boosting (0.78), staking (0.72), and voting (0.76). For the class level metric, the performance outcome did not match other metrics that included voting (0.82), staking (0.82), boosting (0.74), and bagging (0.78) utilizing the AUC-Curve[28].

Arvinder Kaur et al. state that the primary goal of the study was to assess the use of random forest application for error prone class prediction utilising open source software. To perform research, the researcher employed object-oriented metrics using the open-source JEdit programme. According to the experiment results, the precision, recall, F-measure, accuracy RF, and AUC are 74.24%, 72%, 79%, and 0.81, respectively[29].

The YI PENG group, The paper's objective was to evaluate ensemble techniques' quality in software failure prediction using an analytical hierarchal process. The researcher uses 10 publicly available NASA MDP data sets and 13 different performance indicators.

In this paper, Bagging, Boosting, and Staking was an ensemble method. Decision trees provide the greatest results with an accuracy of 92.53% when measured by the AdaBoost performance metric; in this instance, decision trees are the base classifier [/30].

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

Conclusion

Because of its advantages, software-based system development is growing more and more these days compared to earlier years. But before the system is given to end users, quality control must be done. We have a number of quality measurements, including software testing, CMM, and ISO standards, to improve the software quality. These days, software testing plays a bigger and bigger 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 resources to the modules that are prone to errors. This study's primary goal was to evaluate earlier research on software defects that use machine learning techniques, data sets, tools, methodologies, and scientific contributions. We divided the research into three categories: ensemble methods, clustering, and classification. Lastly, a systematic literature review that incorporates books, dissertations, tutorials, and theses can be used to expand on this research.

References

[1] Sandeep D and R. S, "Case Studies of Most Common and Severe Types of Software System Failure " International Journal of Advanced Research in Computer Science and Software Engineering vol. 2, pp. 341-347 August 2012

[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, "Why Software Fails " 2005

[4] Rajkumar G and K.Alagarsamy, "The Most Common Factors For The Failure Of Software Development Project," vol. 11, pp. 74-77, January 2013.

[5] L. J, "Major Causes of Software Project Failures " CROSSTALK The Journal of Defense Software Engineering pp. 9-12, July 1998.

[6] T. Clancy, "The Standish Group Report CHAOS," Project Smart pp. 1-16, 2014.

[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.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

[8] Mikyeong P and E. H, "Software Fault Prediction Model using Clustering Algorithms Determining the Number of Clusters Automatically," International Journal of Software Engineering and Its Applications, vol. 8, pp. 199- 204, 2014.

[9] Harry A, "machine learning capabilities, limitation and implications " Technology Futures Researcher at Nesta 2015.

[10] T. M. Mitchell, "Machine Learning " 1997.

[11] George T, Ioannis K, Ioannis P, and Ioannis V, "Modern Applications of Machine Learning " Proceedings of the 1st 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] L. C. Briand, "Novel Applications of Machine Learning in Software Testing," pp. 3-10, 2008.

[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] Nguyen V, "The Application of Machine Learning Methods in Software Verification and Validation " MSc, Master of Science in Engineering, The University of Texas at Austin, Texas 2010.

[19] Ekbal R, Srikanta P, and V. B, "Software Quality Estimation using Machine Learning:Case-based Reasoning Technique " International Journal of Computer Applications, vol. 58, pp. 43-48, November 2012.

[20] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," Expert Systems with Applications, 2014.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 8, Issue 1, 2019

[21] M. SURENDRA and D. N. GEETHANJALI, "Classification of Defects In Software Using Decision Tree Algorithm," vol. 5, pp. 1332-1340, June 2013.

[22] 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.

[23] 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.

[24] X. Tan, X. Peng, S. Pan, and W. Zhao, "Assessing Software Quality by Program Clustering and Defect Prediction," pp. 244-248, 2011.

[25] Jaspree 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.

[26] 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.