JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Review of Software Defects Prediction Using Machine Learning Approach

[1]Oshin Fernandes, [2]Mr. Dilip Ahirwar
[1]M.Tech Scholar, [2]Assistant Professor
Department of Computer Science and Engineering
Jai Narain College of Technology, Bhopal, India

Abstract : Software defect prediction is a critical aspect of software development and maintenance, as it significantly influences the reliability, efficiency, and quality of software systems. Identifying potential defects early in the software development life cycle (SDLC) reduces costs, saves time, and enhances overall system performance. Machine learning approaches have emerged as powerful tools for predicting software defects by analyzing historical defect data and identifying patterns that indicate problematic areas in the code. This review aims to explore the state-of-the-art machine learning techniques applied to software defect prediction, highlighting their strengths, limitations, and application scenarios. By examining various algorithms, such as decision trees, support vector machines (SVM), neural networks, and ensemble methods, this review provides insights into their effectiveness and suitability for different contexts. The findings of this review aim to serve as a valuable resource for researchers and practitioners seeking to adopt machine learning methods to enhance software quality and streamline development processes.

Index Terms **– Machine Learning, Software defects, Prediction.**

I. INTRODUCTION

Software defects, often referred to as bugs, are inevitable in the software development process and can lead to system failures, performance degradation, or security vulnerabilities[1]. Detecting and rectifying these defects at the earliest stage of the software development life cycle is crucial for minimizing costs and ensuring the delivery of reliable software products. Traditional defect detection methods, such as code reviews and manual testing, are time-consuming and prone to human error, especially in large-scale software systems[2]. The advent of machine learning has revolutionized defect prediction by enabling automated analysis of vast amounts of software data, thus improving accuracy and efficiency in identifying potential problem areas[3].

Machine learning techniques for software defect prediction leverage historical defect datasets and apply algorithms to uncover patterns and correlations between software attributes and defect occurrences[4]. These approaches provide a proactive mechanism to identify modules or components that are likely to contain defects, allowing developers to focus their testing efforts effectively[5]. Commonly used machine learning algorithms in this domain include logistic regression, decision trees, support vector machines, neural networks, and ensemble methods like random forests and gradient boosting. Each of these techniques offers unique advantages in terms of predictive accuracy, computational efficiency, and adaptability to various types of data[6].

One of the primary challenges in software defect prediction is the availability of high-quality datasets that accurately reflect real-world software development scenarios. Imbalanced datasets, where defective modules are significantly outnumbered by non-defective ones, often pose difficulties for machine learning models[7]. Additionally, the selection of relevant features, such as code complexity metrics, process metrics, and change metrics, plays a critical role in enhancing model performance[8]. This review aims to provide a comprehensive analysis of machine learning approaches for software defect prediction, focusing on their strengths, limitations, and practical applications. By exploring the advancements in this field, the review seeks to contribute to the development of more reliable and efficient defect prediction models, ultimately improving software quality and reducing development costs.

II. BACKGROUND

M. Vardhan et al.,[1] The amount of information on web is expanding as the client of the web develops. Presently, Software is sufficiently skilled to replace all that requires human intervention or presence. SDLC (Software Defect LifeCycle) is process that is continued in fostering the Software, which contains thorough testing. To try not to sit around and exertion on Bugs, Software Bug Prediction plays a pivotal part in the beginning stage of improvement in keeping up with and fostering the great nature of Software. Programming Bug Prediction worries with by and large accomplishment as early programming expectation can work

on the quality, dependability, proficiency, decreased cost of Software. We have used different AI approaches on NASA dataset and contrasted them and as of late presented calculation called Catboost. The results achieved, when contrasted and other approaches, CatBoost method rout them for all dataset.

M. A. Shehab et al.,[2] Existing Just-in-Time (JIT) bug prediction techniques are designed to work on single projects. In this paper, we present ClusterCommit, a JIT bug prediction approach geared towards clusters of projects that share common libraries and functionalities. Unlike existing techniques, ClusterCommit trains a machine learning model by combining commits from a set of projects that are part of a larger cluster. Once this model is built, ClusterCommit can be used to detect buggy commits in each of these projects. When applying ClusterCommits to 16 projects that revolve around the Hadoop ecosystem and 10 projects of the Hive ecosystem, the results show that ClusterCommit achieves an F1-score of 73% and MCC of 0.44 for both clusters. These preliminary results are very promising and may lead to new JIT bug prediction techniques geared towards projects that are part of a large cluster.

N. Khanna et al.,[3] In order to reduce developers' time testing the defects in software, this research work has come up with an ensemble learning approach using various machine learning and deep learning classifiers. This empirical study starts with the data collection process. Own dataset has been created by using Understand tool and further it will be applied over sampling techniques. The data set has a binary class output, and thus by applying data over sampling techniques, the number of instances of both classes remain comparable. After that various classifiers are trained with the entire training data set. After this step, the proposed model will predict the outcome that these classifiers would create using the training set and compare them with the actual values of the training dependent variable. After this the information that claim which of the classifiers were able to correctly predict the outcome will be given a data point. After that the neural networks will be trained for each of the classifiers with input variables as training independent variables and output variables as the previously recorded data as to whether a classifier correctly classifies a data point or not. This predicting network would tell us whether the classifier corresponding to that particular network would correctly classify a particular data point or not. To predict the outcome of any new data point, it will be fed to the predicting network and find which classifier would correctly predict the outcome and the aggregated result of the selected classifiers would be reported.

S. A. K, V. Gururaj et al.,[4] The software development life cycle is a long and complicated process. It consists of analysis, design, development, testing and deployment. Defect prediction is the technique of creating models that detect defective systems such as units or classes in the early stages of the process. The major goal of Software Defect Prediction is to detect defects prone in the program and thereby reduce the effort, time and cost involved to the minimum. This paper gives a comprehensive review of all the techniques to approach defect prediction. The PROMISE repository which is a public software defect prediction dataset which is owned by the National Aeronautics and Space Administration (NASA) is used. More than 30 research papers in the domain of software defect prediction were analysed and reviewed.

M. Liang et al.,[5] Long-running software systems tend to exhibit performance degradation and increase failure rate, and the phenomenon is known as software aging. The bugs that cause the aging phenomenon are called Aging-Related Bugs (ARBs), and may bring serious economic loss or even endanger human security. To discover and remove ARBs, ARBs prediction is presented. But ARBs prediction model often needs a large number of training data in order to train a high performance classification model. In practice, the labeled data are rare in many cases. In addition, it is difficult to label all samples manually. Furthermore, there is a serious class imbalance problem in ARBs datasets. In order to address the two problems, we propose a framework named QUIRE-HUE. On the one hand, we use a approach named Active Learning by Querying Informative and Representative Examples (QUIRE) to select a few informative and representative samples to label for training set, which can reduce the cost of labeling and get a high performance classification model. On the other hand, we apply a Hashing-Based Undersampling Ensemble (HUE) by constructing diversified training subspaces for undersampling to alleviate class imbalance problem. A set of experiments are performed on two large open-source projects (MySQL, Linux) with six different machine learning classifiers. We use Balance and AUC as the evaluation metrics. Experimental results indicate that QUIRE-HUE achieves encouraging results. Average AUC and Balance are 0.769 and 0.812 respectively on MySQL dataset, 0.772 and 0.828 respectively on Linux dataset, which significantly outperforms all baseline methods.

Z. S. Alharthi et al.,[6] Software testing is a time-consuming and costly task, as it involves testing all software modules. To minimize the cost and effort of software testing, automatic defect detection can be used to identify the defective modules during the early stages. These aid software testers in detecting the modules that require intensive testing. Therefore, automatically predicting software defects has become a critical factor in software engineering. This paper explores the existing methods and techniques on software defect prediction (SDP) and lists the most popular datasets that are used as benchmarks in SDP. In addition, it discusses the approaches to overcome the class imbalance problem, which usually occurs in the benchmark datasets for SDP problems. This paper can be helpful for researchers in software engineering and other related areas.

T. S. Kumar et al.,[7] Fault tolerance and prediction are some of the interesting quality assurance activities of software engineering. Fault prediction algorithms assist the project manager in identifying the parts of the system that are prone to failure and provide the benefit of reducing the time required to test the entire application. The fault-prediction method attempts to detect faulty software modules so that they are used subsequently in the software development process. There are numerous prediction approaches available in the domain of software engineering for detecting defect-prone blocks. Few of them are stable models, and a few are unstable models, which may not be enough to handle all of the possible scenarios and thus fail to produce the earliest prediction. This paper provides an extensive review of existing predictive models for software defect prediction, identify flaws in existing techniques, and address the need for a new prediction approach in the era of fault prediction. The Classification was commonly used to differentiate among faulty and non-faulty modules. This paper proposes two sub modules for prediction of

faults. One submodule to identify the accuracy are implemented and discussed. To achieve better and more accurate results, we intend to build a machine learning model, which can automate the complete process.

M. N. Uddin et al.,[8] Software Defect Prediction (SDP) method plays a vital role to ensure the software quality by predicting bugs in software development phase. In addition, this technique also assists developers to minimize the maintenance costs. In this paper, we proposed a model as SDP-ML that utilizes machine learning classification techniques to predict the faults in the software. In particular, the model uses three gradient boosting classification frameworks LightGBM (LGB), XGBoost (XGB), CatBoost (CB) for the prediction. The classification was performed on five publically available NASA Promise datasets, viz., CM1, JM1, PC1, KC1, KC2, and validated using ten-fold cross-validation techniques. Moreover, the remarkable evaluation techniques Precision, Recall, F1-score, and Accuracy were employed to evaluate the results.

B. Desai et al.,[9] Software product refers to the software which is developed for a specific requirement. Simultaneously, engineering deals with the development of product using explicit technical fundamentals and methods. The software defect can be predicted in diverse stages in which data is utilized as input and pre-processed, attributes are extracted, and classification is performed. This research work makes the implementation of several classifiers in order to predict the software defect. These classifiers are GNB (gaussian naive bayes), Bernoulli NB, RF (random forest) and MLP (multilayer perceptron) which are employed with the objective of forecasting the software defect. The performance of the software defect is enhanced by developing an ensemble classifier. In the introduced ensemble classifier, the PCA (Principal Component Analysis) algorithm is integrated with class balancing. Python is executed to implement the introduced model. Diverse metrics are considered to analyze the results concerning accuracy, precision and recall.

F. Matloob et al.,[10] Recent advances in the domain of software defect prediction (SDP) include the integration of multiple classification techniques to create an ensemble or hybrid approach. This technique was introduced to improve the prediction performance by overcoming the limitations of any single classification technique. This research provides a systematic literature review on the use of the ensemble learning approach for software defect prediction. The review is conducted after critically analyzing research papers published since 2012 in four well-known online libraries: ACM, IEEE, Springer Link, and Science Direct. In this study, five research questions covering the different aspects of research progress on the use of ensemble learning for software defect prediction are addressed. To extract the answers to identified questions, 46 most relevant papers are shortlisted after a thorough systematic research process.

III. CHALLENGES

Software defect prediction using machine learning approaches faces several challenges, starting with data quality and availability. In many cases, datasets contain imbalances, where defective software instances are much fewer than non-defective ones. This imbalance can bias machine learning models, making them more likely to predict non-defective software. Additionally, missing or incomplete data in features such as code complexity or developer histories can compromise the prediction model's accuracy.

Another key challenge is data labeling and class imbalance. Reliable and accurate labeling of defects often requires manual intervention, which can introduce errors. Moreover, the rarity of software defects makes it hard to train models effectively. The class imbalance between defective and non-defective instances leads to models that fail to properly identify defects, especially when rare events are involved.

Machine learning models, especially complex ones, often struggle with overfitting. Overfitting occurs when a model performs well on the training data but fails to generalize to new data. In the context of software defect prediction, this can result in poor performance when the model is applied to new software projects. Additionally, complex models can be difficult to interpret, which reduces their practical utility for developers who need to understand why a defect was predicted.

Temporal aspects of defect prediction pose another challenge. Defect patterns can evolve over time due to software changes, new features, or bug fixes. Predicting defects accurately throughout the software development lifecycle is difficult because defects might appear at different stages, requiring different prediction models. This dynamic nature of software development requires continuous adaptation of models to maintain their relevance.

Evaluating machine learning models in defect prediction is complicated by the choice of evaluation metrics. Metrics like accuracy, precision, recall, and F1-score can vary significantly depending on the dataset. Selecting the appropriate metric is crucial, especially in scenarios with highly imbalanced data where optimizing for one metric may negatively impact others. Moreover, comparing model performance across different datasets and software systems is often not straightforward.

The continuous nature of software development adds complexity to defect prediction models. As software evolves through frequent updates and modifications, defect prediction models need to be retrained to reflect these changes. Ensuring that the model remains effective as the software changes is resource-intensive and requires constant attention to maintain accuracy over time.

Another significant challenge is the lack of domain-specific data. Many datasets used for software defect prediction come from specific types of software or development environments, limiting the generalizability of the models. Additionally, publicly available datasets may not fully capture the nuances of proprietary or niche software systems, making it difficult to develop models that perform well across diverse contexts.

Finally, scalability and real-time prediction present challenges. Software systems are becoming increasingly large, and defect prediction models must handle vast amounts of data efficiently. Implementing real-time prediction for large-scale software can be computationally expensive and difficult to integrate into existing development workflows. Ensuring that defect prediction models work efficiently in these contexts requires innovative solutions in data processing and model deployment.

IV. CONCLUSION

While machine learning approaches hold great promise for software defect prediction, several challenges must be addressed to fully realize their potential. Issues such as data quality, class imbalance, overfitting, and the dynamic nature of software development hinder the effectiveness of these models. Furthermore, the need for accurate labeling, scalable solutions, and continuous model adaptation poses significant barriers. Despite these challenges, ongoing research and innovation in feature selection, evaluation metrics, and model generalization offer hope for developing more robust, interpretable, and effective defect prediction systems that can be seamlessly integrated into real-world software development processes.

REFERENCES

1. M. Vardhan, K. Banerjee and D. Aggarwal, "A Systematic Approach for the Detection of Software Bug Using Catboost," 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON), 2022, pp. 414-419, doi: 10.1109/COM-IT-CON54601.2022.9850519.
2. M. A. Shehab, A. Hamou-Lhadj and L. Alawneh, "ClusterCommit: A Just-in-Time Defect Prediction Approach Using Clusters of Projects," 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 2022, pp. 333-337, doi: 10.1109/SANER53432.2022.00049.
3. N. Khanna, O. Agarwal and Rahul, "Software Change Prediction using Ensemble Learning on Object Oriented Metrics," 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS), 2022, pp. 1369-1373, doi: 10.1109/ICICCS53718.2022.9788196.
4. S. A. K, V. Gururaj, K. R. Umadi, M. Kumar, S. P. Shankar and D. Varadam, "Comprehensive Survey of different Machine Learning Algorithms used for Software Defect Prediction," 2022 International Conference on Decision Aid Sciences and Applications (DASA), 2022, pp. 425-430, doi: 10.1109/DASA54658.2022.9764982.
5. M. Liang, D. Li, B. Xu, D. Zhao, X. Yu and J. Xiang, "Within-Project Software Aging Defect Prediction Based on Active Learning," 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2021, pp. 1-8, doi: 10.1109/ISSREW53611.2021.00037.
6. Z. S. Alharthi, A. Alsaeedi and W. M. S. Yafooz, "Software Defect Prediction Approaches: A Review," 2021 4th International Conference on Bio-Engineering for Smart Technologies (BioSMART), 2021, pp. 1-6, doi: 10.1109/BioSMART54244.2021.9677869.
7. T. S. Kumar and B. Booba, "A Systematic Study on Machine Learning Techniques for Predicting Software Faults," 2021 IEEE Mysore Sub Section International Conference (MysuruCon), 2021, pp. 133-136, doi: 10.1109/MysuruCon52639.2021.9641090.
8. M. N. Uddin, B. Li, M. N. Mondol, M. M. Rahman, M. S. Mia and E. L. Mondol, "SDP-ML: An Automated Approach of Software Defect Prediction employing Machine Learning Techniques," 2021 International Conference on Electronics, Communications and Information Technology (ICECIT), 2021, pp. 1-4, doi: 10.1109/ICECIT54077.2021.9641218.
9. B. Desai and E. N. Kapoor, "Hybrid Classification Approach for Software Defect Prediction with Feature Reduction and Clustering," 2021 2nd Global Conference for Advancement in Technology (GCAT), 2021, pp. 1-7, doi: 10.1109/GCAT52182.2021.9587763.
10. F. Matloob et al., "Software Defect Prediction Using Ensemble Learning: A Systematic Literature Review," in IEEE Access, vol. 9, pp. 98754-98771, 2021, doi: 10.1109/ACCESS.2021.3095559.