



Software Defect Prediction using Different Machine Learning Technique: A Review

Pragya Kushwah¹, Dr. Sanjeev Kumar Sharma², Dr. Bhagbati Charan Patel

M. Tech. Scholar, Department of Computer Science and Engineering, TIT, Bhopal¹

Professor, Department of Computer Science and Engineering, TIT, Bhopal^{2,3}

Abstract: Software defect prediction is a thriving study area in the realm of software engineering and processing in the IOT-based environment. Defect prediction creates a list of defective source code artifacts so that quality assurance companies may successfully assign limited methods for certifying programming things by investing more effort into the bad source code. Defect prediction can assist estimate maintenance times, which can help with quality assurance, dependability, security, and cost reduction. Many predictions in IOT-based processing environment and business process management and enhancement challenges still exist in defect expectation ponders, and there are various noteworthy concerns. Machine learning software defect prediction is a promising field of software engineering, attracting a great deal of attention from the research community; however, its industry application tends to lag behind academic achievements. Even though the number of machine learning software defect prediction studies validated in the industry is increasing there is still not enough practical focus on the business aspects of the effort that would help bridge the gap between the needs of the industry and academic research.

Index Terms – Software Testing, Machine Learning, Software Defect

I. INTRODUCTION

Defect prediction of software is a vibrant research domain in the field of software engineering environment [1]. The Defect prediction outcomes of the defective source code antiquities with the goal that quality confirmation in organizations can successfully assign restricted means for approving programming items by putting additional exertion on the poor source code [2]. The term defect usually mentions to some problem with the software, either with its exterior performance or with its interior characteristics. A software defect is an error, flaw, failure, or fault in a computer program or system that sources it to crop an improper or unpredicted outcome, or to an act in unintended behaviors [3]. Most advanced software systems beyond restricted personal utilization have become gradually huge and more complex because of the augmented necessity for automation, functions, characteristics, structures, and services. It is almost impossible to entirely prevent or remove defects in such huge complex systems [4].

As the amount of software produces ends up noticeably bigger, defect forecast systems will assume an essential part to help software engineers and additionally accelerate time to a marketplace with more solid software products. Organizations are capitalizing heavily on the operational environment and organizational applications. Software defects in the operational environment are defined as unpredicted interruptions which affect the system productivity and have also an impact on the cost [5]. To minimize the unscheduled interruptions and increase the performance, many defect prediction management techniques are introduced. IT service providers are constantly seeking more efficient procedures and methods to improve the effectiveness and superiority of the process. IT Substructure Library is the most common framework for IT services due to its best management guidelines [6]. It provides the best guidelines on how to manage, develop, and maintain the IT substructure. In addition to this, it also gives guidelines on improving the quality of the IT substructure. Software defects on software systems propose that most of the defects occur during the system up-gradation, during the maintenance task, and maybe sometimes due to the system integration [7]. Most common and agreed causes are insufficient testing or poor testing, flaws in documentation or a poor understanding of the system complexity, system overload, resource exhaustion, and complex defect detection routines [8, 9]. Major reasons that cause the systems to be defective are essentially the complex structure and inter-dependencies of the components. A defect or even a partial defect of one system can cause other systems that depend on it to malfunction. (is problem can create a chain of system failures that propagates until it reaches critical components and causes the system to flop. We first discuss the software defects and what types of defects can be faced using the software. With the extensive use of software systems in the current society, the dissident influence of software defects is also expanding [10]. Different quality assurance (QA) alternatives and interrelated techniques can be utilized in a concerted struggle to efficiently and successfully guarantee their quality. Testing is most of the maximum typically performed quality assurance actions for software. It predicts execution troubles so that underlying reasons may be recognized and fixed. Inspection, on the other hand, directly prevents and modifies software troubles without resorting to implementation. Other (QA) replacements such as official verification, defect prevention, and fault tolerance, address within their approaches. Close inspection of how excellent QA replacements cope with defects can assist one in improved utilization of them for unique applications [11–13]. We can perceive that failures, faults, and errors are diverse features of defects. (e wrong algorithm is smeared in numerous modules and becomes the source of many defects (faults or bugs), and a single fault can cause various failures in repetitive executions [14]. A particular error can create different faults, such as when an incorrect algorithm is trying in several components and that becomes the source of several defects (faults or bugs) and a single

fault can create numerous failures in repetitive modules [15]. Conversely, a similar defect can be caused by many faults, such as an interface or interaction failure including several modules, and a similar fault can be there because of various errors. Programming deformity forecast purposes to intuitively perceive problematic programming modules for effective programming tests popular to expand the brilliance of a product framework in the current period of data innovation [16]. Current defect prediction techniques do not perform well for a complex software system. Predicting defects in complex software is no easy task as it is hard to understand and maintain. Many prediction techniques flop in these systems, and performance of the most of them is compromised [10]. Limitations of machine learning moved the trend toward supervised learning and unsupervised learning. But the fact is that supervised learning has some restrictions to be considered. Such supervised learning requires the labeled datasets which are in the historical form, which is costly and time-consuming. Gathering historical datasets manually, automatic engineering is the waste of time and money.

II. LITERATURE REVIEW

Muhammad Azam et al. [1], in the concept of software engineering, software error prediction plays an important role in improving the quality of software systems, which is one of the most important and costly stages in the software development life cycle. soft. As the use of software systems increases in our daily lives, their dependencies and complexity also increase, creating an environment ripe for defects. Due to the existence of software bugs, the software produces incorrect results and behavior. What's more important than defects is detecting them before they happen. Therefore, detecting (and also predicting) software errors allow software managers to effectively allocate resources to the maintenance and testing phases. In the literature, there are different proposals to predict software errors. In this article, we have performed a comparative analysis of machine learning-based software defect prediction systems by comparing. Experimental results show that the proposed models provide results with suitable accuracy for predicting software errors to improve software quality.

L.-Q. Chen et al. [2], SDP has emerged as an essential component of software testing and ensures the delivery of high-quality software products. Programming deformity forecast is separated into conventional programming imperfection expectation and without a moment to spare programming imperfection forecast (JIT-SDP). In any case, the vast majority of the current programming imperfection expectation systems are somewhat improved, which makes it very hard to furnish designers with more definite reference data. This paper proposes a software defect prediction framework based on heterogeneous feature selection and Nested-Stacking to improve software defect prediction and allocate testing resources efficiently. There are three stages to the framework: evaluation of model classification performance, Nested-Stacking classifier, and data set preprocessing and feature selection. The framework's novel heterogeneous feature selection and nested custom classifiers have the potential to significantly raise software defect prediction accuracy. The AUC and F1-score, two comprehensive evaluation indicators, are used in this paper to demonstrate the classification performance of the model on two software defect data sets (Kamei and PROMISE). The investigation did enormous scope inside project imperfection expectation (WPDP) and cross-project deformity forecast (CPDP). According to the findings, the proposed framework outperforms the baseline models in terms of classification performance when applied to the two types of software defect data sets.

M. Pavana et al. [3], programming shortcoming forecast (SFP) has an essential impact in nature of programming. It aids in the early stages of the software development life cycle (SDLC) in identifying flawed constructs. When the predicted results do not match the actual output, this software defect is also known as a defect. This can be alluding as a blunder, shortcoming, bug in a PC program. The topic of machine learning (ML) deals with the process of creating or designing computer programs that constantly improve their effectiveness at specific tasks through experience. In field of computer programming, AI assumes a vital part and contains various methodologies like test exertion expectation and cost forecast. Software faults prediction (SFP) is the most widely studied of these prediction techniques in software engineering. Machine learning techniques like Naive Bayes (NB), support vector machine (SVM), logistic regression (LR), and random forest (RF) were used to make the predictions in this paper. When selecting highly correlated input variables, a feature selection method like Spearman's rank correlation is used. Dimensionality decrease techniques, for example, LDA and PCA are utilized for diminishing the aspects. These algorithms are evaluated and performed with accuracy, precision, and recall.

A. Al-Nusirat et al. [4], by focusing on defect modules, software defect prediction is a practical way to increase the quality, efficiency, and cost of software testing. The dataset of programming imperfection expectation normally has a class unevenness issue with not many deficient modules contrasted with non-faulty modules. The Neural Network suffers as a result of this circumstance, which can result in inaccurate fitting and overfitting. Engineered Minority Over-inspecting Procedure (Destroyed) is one of the famous strategies that can take care of the issue of class irregularity. However, the user must determine the hyperparameters for both SMOTE and Neural Networks prior to modeling. In this review, we applied the Brain Organizations Based Destroyed, a blend of Brain Organization and Destroyed with each hyperparameter of Destroyed and Brain Organization that are upgraded utilizing irregular pursuit to take care of the class lopsidedness issue in the six NASA datasets. The outcomes utilize a 5*5 crossapproval show that expands Bal by 25.48% and Review by 45.99% contrasted with the first Brain Organization. SMOTE based on neural networks and SMOTE based on "traditional" machine learning are also compared in terms of performance. In terms of average rank, the neural network-based SMOTE comes out on top.

R. Bahaweres et al. [5], various programming imperfection expectation models have been proposed to work on the nature of programming throughout recent many years. An inexorably famous methodology is to utilize AI. There are two types of these strategies: supervised methods, in which the training data must be labeled, whether they are faulty or not, and unsupervised methods, in which the data do not require labeling. Regulated forecast models prevail. Notwithstanding, by and by it is frequently hard to gather deformity order names to prepare a Managed Imperfection Forecast (SDP) model. Consequently, Unsupervised Defect Prediction (UnSDP) models have begun to gain attention in recent years.

Y. Qiu et al. [6], many software engineering tasks rely heavily on hand-designed software features, e.g., error prediction, vulnerability detection, software requirements, code review, and malware detection. Previous solutions to these tasks often directly used hand-generated features or feature selection techniques for classification or regression, which often led to suboptimal results due to lack of Strong representation of hand-generated features. To solve the above problem, in this paper, we apply just-in-time software defect prediction (JIT-SDP), which is a typical task that relies on handcrafted features, e.g. to exploit new possible solutions. We propose a new model, named neural forest (NF), which uses deep neural networks and decision forests to build an overall system to automatically discover robust feature representations. strongly used for further classification. NF first uses a deep neural network to learn new feature representations from hand-generated features. Then, a decision forest is connected behind the neural network to perform classification and also guide object representation learning. NF mainly aims to solve the difficult problem of end-to-end combination of the two different worlds of neural networks and decision forests. Compared with previous state-of-the-art defect prediction tools and five benchmarks designed based on six well-known benchmarks for intra- and inter-project defect prediction, NF achieves significantly better results. The proposed NF model is a general model for classification problems based on handcrafted features.

A. Alsaeedi et al. [7], finding and addressing defects that could be anticipated under a variety of circumstances ahead of schedule is an essential goal of software development. Because so many aspects of software development are carried out by individuals, there is a possibility that numerous software bugs will emerge during the process, resulting in disappointments in the not-too-distant future. Hence, the forecast of programming defects in the principal stages has turned into an essential interest in the field of computer programming. Different programming imperfection expectation (SDP) approaches that depend on programming measurements have been proposed over the most recent twenty years. It is known that classifiers based on bagging, support vector machines (SVM), decision trees (DS), and random forests (RF) are effective at predicting defects. On ten NASA datasets, these supervised machine learning and ensemble classifiers are examined and contrasted in this paper. The experimental results demonstrated that, in the vast majority of instances, RF outperformed the other classifiers in terms of performance.

C. Manjula et al. [8], numerous methods, including data mining and machine learning, have been developed for the early prediction of software defects. Still early forecast of deformities is a difficult errand which should be tended to and can be improved by getting higher order pace of imperfection expectation. We present a hybrid strategy that combines a genetic algorithm (GA) for feature optimization with a deep neural network (DNN) for classification in order to address this issue. A better variant of GA is integrated which incorporates another strategy for chromosome planning and wellness capability calculation. DNN method is additionally ad libbed utilizing versatile auto-encoder which gives better portrayal of chosen programming highlights. Through case studies, the proposed hybrid approach's increased efficiency as a result of using an optimization technique is demonstrated. Using the MATLAB tool, an experimental study on software defect prediction is conducted taking into account the PROMISE dataset. In this review, we have involved the proposed novel technique for characterization and imperfection expectation. According to a comparison study, the proposed method for predicting software defects outperforms other methods with an accuracy of 97.82 percent for the KC1 dataset, 97.59 percent for the CM1 dataset, 97.96 percent for the PC3 dataset, and 98.00 percent for the PC4 dataset.

R. Jayanthi et al. [9], the software industry strives to improve software quality through consistent bug prediction, bug removal, and module fault prediction. Because of its significant involvement in software industries, this field has attracted researchers. Different methods have been introduced for programming deformity expectation. Data-mining with machine learning has been suggested as an important paradigm for software bug prediction by recent studies. The classification accuracy of state-of-the-art software defect prediction tasks is one of a number of problems. Be that as it may, programming deformity datasets are imbalanced in nature and known shortcoming inclined because of its enormous aspect. A combined strategy for the prediction of software bugs and software defects is presented here as a solution to this problem. Proposed approach conveys an idea of element decrease and man-made consciousness where highlight decrease is completed by notable rule part examination (PCA) conspire which is additionally improved by consolidating most extreme probability assessment for blunder decrease in PCA information reproduction. At last, brain network based characterization procedure is applied which shows forecast results. A structure is planned and carried out on NASA programming dataset where four datasets i.e., KC1, PC3, PC4 and JM1 are considered for execution investigation utilizing MATLAB reenactment device. A broad exploratory review is performed where disarray, accuracy, review, characterization precision and so on parameters are calculated and compared to existing methods for predicting software defects. Trial concentrate on demonstrates the way that proposed approach can give better execution to programming deformity expectation.

Kondo et al. [10], numerous programming assignments vigorously depend available created programming highlights, for example , imperfection expectation, weakness revelation, programming prerequisites, code audit, and malware recognition. Past answers for these undertakings typically straightforwardly utilize the hand-created highlights or component choice methods for grouping or relapse, which as a rule prompts sub-standard outcomes because of their absence of strong portrayals of the hand-made highlights. This paper uses the effort-aware just-in-time software defect prediction (JIT-SDP), a typical hand-crafted feature-based task, as an example to explore new potential solutions to the aforementioned issue. Neural forest (NF) is a new model we propose that builds on deep neural networks and decision forests to create a comprehensive system for automatically exploring powerful feature representations used in the following classification. In the beginning, NF makes use of a deep neural network to learn brand-new feature representations from manually crafted features. Following the neural network, a decision forest is connected to perform classification and direct feature representation learning simultaneously. NF predominantly targets taking care of the difficult issue of consolidating the two unique universes of brain organizations and choice timberlands in a start to finish way. For within- and cross-project defect prediction, NF achieves significantly better results than previous state-of-the-art defect predictors and five designed baselines on six wellknown benchmarks. The classification problems that rely on the hand-crafted features can be applied to the proposed NF model.

III. MACHINE LEARNING

Uproarious information is available in the heap of substance that will be identified through the anomaly strategies. The information can be spatial or can be a transient method spatial connected with the geological conditions and worldly connected with the time perspectives [14, 15]. The principle point of exception identification is to deal with the loud information that is introduced in the heap of text. Different methods for recognizing abnormalities in Text are specified in below:

Learning

The main property of an ML is its capability to learn. Learning or preparing is a procedure by methods for which a neural system adjusts to a boost by making legitimate parameter modifications, bringing about the generation of wanted reaction. Learning in an ML is chiefly ordered into two classes as [16].

- Supervised learning
- Unsupervised learning

Supervised Learning

Regulated learning is two stage forms, in the initial step: a model is fabricated depicting a foreordained arrangement of information classes or ideas. The model developed by investigating database tuples portrayed by traits. Each tuple is expected to have a place with a predefined class, as dictated by one of the qualities, called to have a place with a reclassified class, as controlled by one of the traits called the class name characteristic. The information tuple are dissected to fabricate the model all things considered from the preparation dataset.

Unsupervised learning

It is the kind of learning in which the class mark of each preparation test isn't knows, and the number or set of classes to be scholarly may not be known ahead of time. The prerequisite for having a named reaction variable in preparing information from the administered learning system may not be fulfilled in a few circumstances.

Data mining field is a highly efficient techniques like association rule learning. Data mining performs the interesting machine-learning algorithms like inductive-rule learning with the construction of decision trees to development of large databases process. Data mining techniques are employed in large interesting organizations and data investigations. Many data mining approaches use classification related methods for identification of useful information from continuous data streams.

Nearest Neighbors Algorithm

The Nearest Neighbor (NN) rule differentiates the classification of unknown data point because of closest neighbor whose class is known. The nearest neighbor is calculated based on estimation of k that represents how many nearest neighbors are taken to characterize the data point class. It utilizes more than one closest neighbor to find out the class where the given data point belong termed as KNN. The data samples are required in memory at run time called as memory-based technique. The training points are allocated weights based on their distances from the sample data point. However, the computational complexity and memory requirements remained key issue. For addressing the memory utilization problem, size of data gets minimized. The repeated patterns without additional data are removed from the training data set.

Naive Bayes Classifier

Naive Bayes Classifier technique is functioned based on Bayesian theorem. The designed technique is used when dimensionality of input is high. Bayesian Classifier is used for computing the possible output depending on the input. It is feasible to add new raw data at runtime. A Naive Bayes classifier represents presence (or absence) of a feature (attribute) of class that is unrelated to presence (or absence) of any other feature when class variable is known. Naïve Bayesian Classification Algorithm was introduced by Shinde S.B and Amrit Priyadarshi (2015) that denotes statistical method and supervised learning method for classification. Naive Bayesian Algorithm is used to predict the heart disease. Raw hospital dataset is employed. After that, the data gets preprocessed and transformed. Finally by using the designed data mining algorithm, heart disease was predicted and accuracy was computed.

Support Vector Machine

SVM are used in many applications like medical, military for classification purpose. SVM are employed for classification, regression or ranking function. SVM depends on statistical learning theory and structural risk minimization principal. SVM determines the location of decision boundaries called hyper plane for optimal separation of classes as described in figure 3. Margin maximization through creating largest distance between separating hyper plane and instances on either side are employed to minimize upper bound on expected generalization error. Classification accuracy of SVM not depends on dimension of classified entities. The data analysis in SVM is based on convex quadratic programming. It is expensive as quadratic programming methods need large matrix operations and time consuming numerical computations.

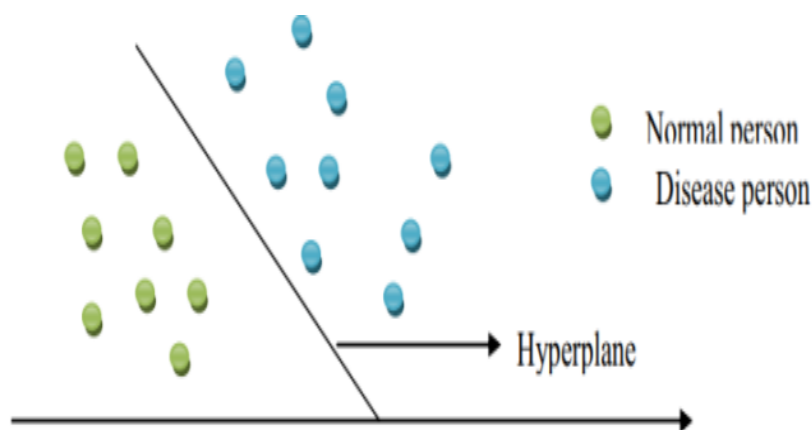


Figure 1: Support Vector Classification

IV. CONCLUSION

Generally, each software defect is essential regarding quality, reliability, security, and cost-effectiveness. Defect prediction help in predicting the maintenance times, which counteract quality assurance, reliability, security richness, and reduce costs. (is study evaluated and analyzed different SSL methodologies in which the Extended Random Forest (extRF) technique is used for the defective system prediction. (e Extended Random Forest (extRF) technique is an extended form of the Random Forest approach, which is a supervised learning approach to semi-supervised learning getting the hang of refining every arbitrary tree given an individual-training worldview. A boosting procedure is conferred, and a weighted mixture of irregular trees creates the final forecast results. After analyzing the experimental results of this study, we can conclude that sampling with Semi-Supervised Learning (SSL) and active learning can attain improved prediction presentation than sampling with predictable ML techniques. A sample might comprise abundant information that a predictable ML learner has already educated very well; however, it might comprise minor information that the learner requires for increasing the present prediction accurateness. In future work, this study can be extended to incorporate the research on the legality of our evaluation and its comparison with the other proposed models for defect prediction.

REFERENCES

- [1] Muhammad Azam, Muhammad Nouman, Ahsan Rehman Gill, "Comparative Analysis of Machine Learning techniques to Improve Software Defect Prediction", *Journal of Computing & Information Sciences (KJCIS)* Volume 5, Issue 2, pp. 41-66, 2022.ue 2
- [2] L.-Q. Chen, C. Wang, and S.-L. Song, "Software defect prediction based on nested-stacking and heterogeneous feature selection", *Complex Intell. Syst.*, vol. 8, no. 4, pp. 3333–3348, 2022.
- [3] M. Pavana, L. Pushpa, and A. Parkavi, "Software fault prediction using machine learning algorithms," in *Proc. Int. Conf. Adv. Elect. Comput. Technol.*, 2022, pp. 185–197
- [4] A. Al-Nusirat, F. Hanandeh, M. K. Kharabsheh, M. AlAyyoub, and N. Al-Dhfairi, "Dynamic detection of software defects using supervised learning techniques," *Int. J. Commun. Netw. Inf. Secur.*, vol. 11, no. 1, pp. 185–191, Apr. 2022.
- [5] Tiapride, Jaray's, Chakri Klan Tantithamthavorn, Hao Khan Dam, and John Grundy, "An Empirical Study of ModelAgnostic Techniques for Defect Prediction Models." *IEEE Transactions on Software Engineering*, vol. 48, No. 1, pp. 166–85, 2022.
- [6] R. Bahaweres, F. Agustian, I. Hermadi, A. Suroso, and Y. Arkeman, "Software defect prediction using neural network based SMOTE", in *Proc. 7th Int. Conf. Electr. Eng., Comput. Sci. Informat. (EECSI)*, pp. 71–76, 2020.
- [7] Y. Qiu, Y. Liu, A. Liu, J. Zhu, and J. Xu, "Automatic feature exploration and an application in defect prediction," *IEEE Access*, vol. 7, pp. 112097–112112, 2019.
- [8] A. Alsaedi and M. Z. Khan, "Software defect prediction using supervised machine learning and ensemble techniques: A comparative study", *J. Softw. Eng. Appl.*, vol. 12, no. 5, pp. 85–100, 2019.
- [9] C. Manjula and L. Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics," *Cluster Comput.*, vol. 22, no. S4, pp. 9847–9863, Jul. 2019.
- [10] R. Jayanthi and L. Florence, "Software defect prediction techniques using metrics based on neural network classifier," *Cluster Comput.*, vol. 22, no. S1, pp. 77–88, Jan. 2019.
- [11] Kondo, Masanari, Cor-Paul Beemer, Yasutaka Kamei, Ahmed E. Hassan, and Osamu Mizuno, "The Impact of Feature Reduction Techniques on Defect Prediction Models." *Empirical Software Engineering*, vol. 24, no. 4, pp. 1925–63, 2019.
- [12] Manjula, C., and Lilly Florence, "Deep Neural Network Based Hybrid Approach for Software Defect Prediction using Software Metrics", *Cluster Computing* 22:9847–63, 2019.
- [13] A. Hammouri, M. Hammad, M. Alnabhan, and F. Alsarayrah, "Software bug prediction using machine learning approach," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 2, pp. 78–83, 2018.
- [14] N. Kalaivani and R. Beena, "Overview of software defect prediction using machine learning algorithms," *Int. J. Pure Appl. Math.*, vol. 118, pp. 3863–3873, Feb. 2018.
- [15] M. A. Memon, M.-U.-R. Magsi, M. Memon, and S. Hyder, "Defects prediction and prevention approaches for quality software development," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 8, pp. 451–457, 2018.
- [16] E. Naresh and S. P. Shankar, "Comparative analysis of the various data mining techniques for defect prediction using the NASA MDP datasets for better quality of the software product," *Adv. Comput. Sci. Technol.*, vol. 10, no. 7, pp. 2005–2017, 2017