International Journal of Research and Development in Applied Science and Engineering (IJRDASE)
ISSN: 2454-6844

Artificial Intelligence Approaches for Predicting Software Defects: A Comprehensive Review

Sanjay Kumar, Sameer Awasthi
Dept of Computer Science & Engineering
Bansal Institute of Engineering and Technology, India,
callsanjay7376@gmail.com, Sameer.awasthi@gmail.com

Abstract- **Software faults remain a significant challenge in the field of software engineering. These faults, which are essentially defects, can lead to failures in executable software products. Identifying the root causes of such defects early in the development lifecycle—and recognizing areas within the software process that may require attention—can result in substantial savings in time, cost, and effort. The ability to predict software fault-proneness at early stages of development can enhance the planning, management, and execution of software projects. This paper presents a comprehensive survey of research conducted over recent years, focusing on the evolution of various prediction techniques aimed at addressing issues related to software defects.**

Keywords: Software Defects, RMSE, ANFIS, MDP.

1. Introduction

Software defects, also known as bugs or faults, are errors in software systems that can cause incorrect or unexpected behavior. Identifying and fixing these defects is essential for delivering high-quality, reliable software. Manual testing and traditional static code analysis techniques are often inadequate due to the complexity and size of modern software systems. Consequently, there has been a growing interest in employing AI techniques to predict software defects efficiently.

Artificial intelligence, particularly machine learning and deep learning, has proven effective in learning patterns from historical defect data and making accurate predictions. This paper reviews the state-of-the-art AI techniques used in software defect prediction, providing a comprehensive understanding of their development, application, and impact. Faults in software systems continue to be a major problem [4]. Software bug is an error, mistake, flaw, failure, or fault in a computer program that prevents it from behaving as intended (e.g., producing an incorrect result) [5]. A software fault is a defect that causes software failure in an executable product. In software engineering, the non-conformance of software to its requirements is commonly called a bug. Most bugs occur from mistakes and errors made by persons in either a program's design or its source code, a few are caused by compilers producing incorrect code. Knowing the causes of possible defects as well as identifying general software process areas that may need attention from the initialization of a project could save money, time and work. The possibility of early estimating the potential faultiness of software could help on planning, controlling and executing software development activities.

A wide range of prediction models have been proposed. Complexity and size metrics have been used in an attempt to predict the number of defects a system will reveal in operation or testing. Reliability models have been developed to predict failure rates based on the expected operational usage profile of the system. Information from defect detection and the testing process has been used to predict defects. The maturity of design and testing processes have been advanced as ways of reducing defects. Recently large complex multivariate statistical models have been produced in an attempt to find a single complexity metric that will account for defects. This paper provides a *critical* review of the various work carried out in this field with the purpose of identifying future avenues of research.

2. Traditional Machine Learning Approaches

Early research in software defect prediction primarily utilized traditional machine learning algorithms, which include:

- **Decision Trees (DT):** Easy to interpret and implement, decision trees have been widely used due to their simplicity and effectiveness.
- **Support Vector Machines (SVM):** Known for high accuracy, especially in binary classification tasks.
- **Naive Bayes (NB):** A probabilistic classifier that is computationally efficient.
- **k-Nearest Neighbors (k-NN):** A non-parametric method that predicts based on the closest training examples.
- **Random Forest (RF):** An ensemble learning method that enhances accuracy by combining multiple decision trees.

These models rely on historical metrics such as code complexity, churn, and developer activity. While they offer reasonable prediction performance, they often require feature engineering and may struggle with non-linear relationships in data.

3. Deep Learning Techniques

With advancements in computational power, deep learning has been increasingly applied to software defect prediction. Notable deep learning models include:

- **Artificial Neural Networks (ANN):** Capable of capturing complex patterns in data through multiple hidden layers.
- **Convolutional Neural Networks (CNN):** Though primarily used in image processing, CNNs have been adapted to handle structured code data.
- **Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM):** Effective for sequential data and code trace analysis.

Deep learning models generally outperform traditional machine learning methods in terms of accuracy and generalization, especially with large and complex datasets. However, they require substantial computational resources and large labeled datasets.

4. Ensemble and Hybrid Methods To further improve prediction accuracy and robustness, researchers have proposed ensemble and hybrid models:

- **Bagging and Boosting:** Techniques like AdaBoost and Gradient Boosting combine multiple weak learners to form a strong predictor.
- **Stacking:** Integrates multiple classifiers by training a meta-classifier on their outputs.
- **Hybrid Models:** Combine machine learning with optimization algorithms (e.g., Genetic Algorithms, Particle Swarm Optimization) or fuzzy logic to enhance model performance and interpretability.

These approaches aim to leverage the strengths of different algorithms and mitigate their weaknesses.

5. Common Datasets and Evaluation Metrics

Several publicly available datasets are widely used for evaluating software defect prediction models, such as:

- PROMISE repository
- NASA MDP datasets
- Eclipse datasets

Evaluation metrics typically include:

- Precision, Recall, F1-score
- Area Under the ROC Curve (AUC)
- Matthews Correlation Coefficient (MCC)
- Accuracy

These metrics help in assessing the model's ability to correctly identify defective modules.

6. Challenges in Software Defect Prediction

Despite significant progress, several challenges persist:

- **Data Imbalance:** Most software datasets are highly imbalanced, with far fewer defective modules.
- **Feature Selection:** Identifying relevant features significantly affects model performance.
- **Generalization:** Models trained on one project may not perform well on others.
- **Interpretability:** Deep learning models often function as black boxes, making it difficult to explain predictions.

7. Review of Literature

Bibi S., Tsoumakas G., Stamelos I., Vlahavas I.(2006) applied a machine learning approach to the problem of estimating the number of defects called Regression via Classification (RvC). To evaluate this approach a comparative experimental study of the effectiveness of several machine learning algorithms in a software dataset was performed. The data was collected by Pekka Forselious and involves applications maintained by a bank of Finland. It was seen that the success of the method is that it provides a framework for discovering potential causes of faults that are not profound like the one that implies that applications for deposit organizations are fault-prone.

Norman Fenton et.al.(1999), have described a probabilistic model for software defect prediction. The aim here is to produce a single model to combine the diverse forms of, often causal, evidence available in software development in a more natural and efficient way than done previously. Here a *critical* review of numerous software metrics and statistical models and the state-of-the art has been carried out. The use of subjective judgements of experienced project managers to build the probability model and use this model to produce forecasts about the software quality throughout the development life cycle has been discussed. This model can not only be used for assessing ongoing projects, but also for exploring the possible effects of a range of software process improvement activities.

Ahmet Okutan, et.al.(2012), proposed a novel method using Bayesian networks to explore the relationships among software metrics and defect proneness. Nine data sets from Promise data repository has been used and show that RFC, LOC, and LOCQ are more effective on defect proneness. In addition to the metrics used in Promise data repository, two more metrics, i.e. NOD for the number of developers and LOCQ for the source code quality has been proposed. At the end of modelling, the usefulness of the marginal defect proneness probability of the whole software system, the set of most effective metrics, and the influential relationships among metrics and defectiveness has been deduced.

Mrinal Singh Rawat et. al.(2012), identified causative factors which in turn suggest the remedies to improve software quality and productivity. They showed how the various defect prediction models are implemented resulting in reduced magnitude of defects. They presented the use of various machine learning techniques for the software fault prediction problem. The unfussiness, ease in model calibration, user acceptance and prediction accuracy of these quality estimation techniques demonstrate its practical and applicative magnetism. These modeling systems can be used to achieve timely fault predictions for software components presently under development, providing valuable insights into their quality. The software quality assurance team can then utilize the predictions to use available resources for obtaining cost effective reliability enhancements.

Supreet Kaur, et.al. (2012), studied the performance of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is evaluated for Fault prediction in Java based Object Oriented Software systems and C++ language based software components. Here, the metric based approach is used for prediction. In case of Java based dataset named as KC3, first, thirty nine metrics are used and later the worth of a subset of attributes is calculated and the number of metrics are reduced to eight. When the worth of an attribute by computing the value of the chi-squared statistic with respect to the class is evaluated, it was seen that that the BRANCH_COUNT and maxHALSTEAD_VOLUME are highest rank metrics for fault prediction in case of Java and C++ based fault prediction dataset respectively.

Xiao-dong Mu et. al.,(2012), in their work to improve the accuracy of software defect prediction, a coevolutionary algorithm based on the competitive organization is put forward for software defect prediction. During this algorithm, firstly, competition mechanism is introduced to organization coevolutionary algorithm. Then, three evolution operators which are reduced operator, allied operators and disturbed

operators are developed for evolution of population. And competition is considered for calculate the fitness function. When the algorithm applied into software defect prediction, it improves the accuracy of software prediction through increases the diversity of population.

N Fenton, et. al. (2008), in their work reviewed the use of Bayesian networks (BNs) in predicting software defects and software reliability. The approach allows analysts to incorporate causal process factors as well as combine qualitative and quantitative measures, hence overcoming some of the wellknown limitations of traditional software metrics methods. Using such 'dynamic discretization' algorithms results in significantly improved accuracy for defects and reliability prediction type models.

Jie Xu, et. al. (2010), used several statistical techniques together with machine learning method verify the effectiveness of software metrics. Moreover, a neuro-fuzzy approach is adopted to improve the accuracy of the estimation model. This procedure is carried out based on data from the ISBSG repository to present its empirical value.

Manu Banga, (2013), here a new computational intelligence sequential hybrid architectures involving Genetic Programming (GP) and Group Method of Data Handling (GMDH) viz. GPGMDH have been discussed. Besides GP and GMDH, a host of techniques on the ISBSG dataset has been tested. The proposed GP- GMDH and GMDH-GP hybrids outperformed all other stand-alone and hybrid techniques. It is concluded that the GPGMDH or GMDH-GP model is the best model among all other techniques for software cost estimation.

Mohamad Mahdi Askari and Vahid Khatibi Bardsiri (2014) used multilayer neural network method in order to improve and increase generalization capability of learning algorithm in predicting software defects. In order to solve the existing problems, a new method is proposed by developing new learning methods based on support vector machine principles and using evolutionary algorithms. The proposed method prevents from overfitting issue and maximizes classification margin. Efficiency of the proposed algorithm has been validated against 11 machine learning models and statistical methods within 3 NASA datasets. Results reveal that the proposed algorithm provides higher accuracy and precision compared to the other models.

Kamaljit Kaur (2012) presented the application of the neural network for the identification of Reusable Software modules in Oriented Software System. Metrics are used for the structural analysis of the different procedures. The values of Metrics will become the input dataset for the neural network systems. Training Algorithm based on Neural Network is experimented and the results are recorded in terms of Accuracy, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Hence the proposed model can be used to improve the productivity and quality of software development.

Mrs.Agasta Adline, Ramachandran. M(2014) Predicting the fault-proneness of program modules when the fault labels for modules are unavailable is a challenging task frequently raised in the software industry. They attempted to predict the fault–proneness of a program modules when fault labels for modules are not present. Supervised techniques like Genetic algorithm based software fault prediction approach for classification has been proposed.

Karpagavadivu.K, et.al. (2012) analyzed the performance of various techniques used in software fault prediction. And also described some algorithms and its uses. They found that the aim of the fault prone module prediction using data mining is to improve the quality of software development process. By using this technique, software manager effectively allocate resources. The overall error rates of all techniques are compared and the advantages of all methods were analyzed.

Xiaoxing Yang, et.al. (2014) introduced a learning-to-rank approach to construct software defect prediction models by directly optimizing the ranking performance. They built the model on previous work, and further studied whether the idea of directly optimizing the model performance measure can benefit software defect prediction model construction. The work includes two aspects: one is a novel application of the learning-to-rank approach to real-world data sets for software defect prediction, and the other is a comprehensive evaluation and comparison of the learning-to-rank method against other algorithms that have been used for predicting the order of software modules according to the predicted number of defects. Our empirical studies demonstrate the effectiveness of directly optimizing the model performance measure for the learning-to-rank approach to construct defect prediction models for the ranking task.

Ahmet Okutan1 and Olcay Taner Yıldız, (2013) proposed a new kernel method to predict the number of defects in the software modules (classes or files). The proposed method is based on a pre-computed kernel matrix which is based on the similarities among the modules of the software system. Novel kernel method with existing kernels in the literature (linear and RBF kernels) has been compared and show that it achieves comparable results. Furthermore, the proposed defect prediction method is also comparable with some existing famous defect prediction methods in the literature i.e. linear regression and IBK. It was seen that prior to test phase or maintenance, developers can use the proposed method to easily predict the most defective modules in the software system and focus on them primarily rather than testing each and every module in the system. This can decrease the testing effort and the total project cost automatically.

Yajnaseni Dash, Sanjay Kumar Dubey, (2012) aimed to survey various research methodologies proposed to predict quality of OO metrics by using neural network approach. The application of artificial neural networks is an efficient method to estimate maintainability in object oriented system. It was seen that among the different soft computing techniques ANN possesses advantages of predicting the software maintenance effort by minimal computation. It can be used as a predictive

model because of its incredible representation techniques and ability to perform complicated functions.

Ms. Puneet Jai Kaur, Ms. Pallavi, (2013) discussed data mining techniques that are association mining, classification and clustering for software defect prediction. This helps the developers to detect software defects and correct them. Unsupervised techniques may be used for defect prediction in software modules, more so in those cases where defect labels are not available.

3. Conclusion:

Artificial intelligence has significantly advanced the field of software defect prediction by providing powerful tools for analyzing complex software data. This review has outlined the progression from traditional machine learning techniques to deep learning and hybrid approaches, evaluating their effectiveness and applicability. As software systems continue to grow in size and complexity, AI-driven solutions will play an increasingly vital role in ensuring software reliability and quality.

References:

[1]. Parvinder S. Sandhu, Sunil Khullar, Satpreet Singh, Simranjit K. Bains, Manpreet Kaur, Gurvinder Singh, "A Study on Early Prediction of Fault Proneness in Software Modules using Genetic Algorithm", World Academy of Science, Engineering and Technology, 2010, pp. 648-653.
[2]. http://puretest.blogspot.com/2009/11/1.html
[3]. Bibi S., Tsoumakas G., Stamelos I., Vlahavas I.(2006), "Software Defect Prediction Using Regression via Classification", IEEE International Conference on Computer Systems and Applications, pp.330 - 336,
[4]. Norman Fenton, Paul Krause and Martin Neil, (1999), "A Probabilistic Model for Software Defect Prediction", For submission to IEEE Transactions in Software Engineering.
[5]. Ahmet Okutan, Olcay Taner Yıldız,(2012) "Software defect prediction using Bayesian networks", Empir Software Eng (2014) 19:154–181 © Springer Science, Business Media, LLC.
[6]. Mrinal Singh Rawat, Sanjay Kumar Dubey,(2012) "Software Defect Prediction Models for Quality Improvement: A Literature Study", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, pp 288-296.
[7]. Supreet Kaur, and Dinesh Kumar, "Software Fault Prediction in Object Oriented Software Systems Using Density Based Clustering Approach", International Journal of Research in Engineering and Technology (IJRET) Vol. 1 No. 2 March, 2012 ISSN: 2277-4378
[8]. Xiao-dong Mu, Rui-hua Chang, Li Zhang, "Software Defect Prediction Based on Competitive Organization CoEvolutionary Algorithm", Journal of Convergence Information Technology(JCIT) Volume7, Number5, 2012.
[9]. N. Fenton and M. Neil (2008) "Using Bayesian networks to predict software defects and reliability", Proc. IMechE Vol. 222 Part O: J. Risk and Reliability, pp 702-712.
[10]. Jie Xu, [2]Danny Ho and [1]Luiz Fernando Capretz, "AN EMPIRICAL STUDY ON THE PROCEDURE TO DERIVE SOFTWARE QUALITY ESTIMATION MODELS", International journal of computer science & information Technology (IJCSIT) Vol.2, No.4, 2010.
[11]. Manu Banga, "Computational Hybrids Towards Software Defect Predictions", International Journal of Scientific Engineering and Technology Volume 2 Issue 5, pp : 311-316, 2013.
[12]. Mohamad Mahdi Askari and Vahid Khatibi Bardsiri (2014), " Software Defect Prediction using a High Performance Neural Network", International Journal of Software Engineering and Its Applications Vol. 8, No. 12 (2014), pp. 177-188.
[13]. Kamaljit Kaur (2012), "ANALYSIS OF RESILIENT BACK-PROPAGATION FOR IMPROVING SOFTWARE PROCESS CONTROL" International Journal of Information Technology and Knowledge Management July-December 2012, Volume 5, No. 2, pp. 377-379.
[14]. Mrs.Agasta Adline, Ramachandran. M(2014), "Predicting the Software Fault Using the Method of Genetic Algorithm", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Special Issue 2,, pp 390-398.
[15]. Karpagavadivu.K, et.al. (2012), "A Survey of Different Software Fault Prediction Using Data Mining Techniques Methods", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 1, Issue 8, pp 1-3.
[16] Anurag et. al., "Load Forecasting by using ANFIS", International Journal of Research and Development in Applied Science and Engineering, Volume 20, Issue 1, 2020
[17] Raghawend, Anurag, "Detect Skin Defects by Modern Image Segmentation Approach, Volume 20, Issue 1, 2020
[18]. Yajnaseni Dash, Sanjay Kumar Dubey, (2012), " Quality Prediction in Object Oriented System by Using ANN: A Brief Survey", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 2,, pp.1-6.
[19]. Ms. Puneet Jai Kaur, Ms. Pallavi, (2013), " Data Mining Techniques for Software Defect Prediction", International Journal of Software and Web Sciences