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

A Brief Review on Software Quality Prediction using Artificial Intelligence

Manish Kumar Maurya, Dr. Rohitashwa Pandey
Department of Computer Science and engineering,
Bansal Institute of Technology, Lucknow, India,
manish22maurya@gmail.com

Abstract: Software quality prediction has emerged as a critical area of research and practice within the software engineering community. The ability to predict the quality of software systems before they are fully developed can significantly reduce the cost of software development and maintenance. This review paper provides an overview of the state-of-the-art in software quality prediction, including the methodologies, models, and tools that have been developed to predict various aspects of software quality. We discuss the challenges and opportunities in this field, highlighting the need for more accurate and reliable prediction models that can adapt to the rapidly changing landscape of software development.

Keywords: Attribute selection, Defect Prediction, Software Quality, Defect Detection.

1. Introduction

Software quality is a multifaceted concept that encompasses various attributes such as reliability, maintainability, performance, and security. Predicting the quality of software before it is released can help developers identify potential issues early in the development process, leading to more robust and reliable software systems. The goal of software quality prediction is to estimate these quality attributes based on available data, such as code metrics, historical data, and user feedback.
Software quality is a multifaceted concept that encompasses the overall value and performance of software systems. It is pivotal in determining the success and reliability of software applications across various industries. At its core, software quality refers to how well a software product meets its requirements and satisfies the needs of its users. It involves a comprehensive assessment of various attributes, including functionality, reliability, usability, efficiency, maintainability, and portability. Ensuring high software quality is crucial because it directly impacts user satisfaction, operational efficiency, and the long-term sustainability of software products.

One of the primary aspects of software quality is functionality, which measures how accurately and completely the software performs the tasks it is intended to perform. This includes the correctness of the software's outputs, the accuracy of its calculations, and its ability to handle edge cases and errors gracefully. Functionality is often evaluated through rigorous testing processes, such as unit testing, integration testing, and system testing, to identify and rectify defects before the software is deployed.

Reliability is another critical dimension of software quality. It refers to the software's ability to operate consistently and correctly over time, without failures. Reliable software minimizes downtime and ensures continuous availability, which is especially important for mission-critical applications. Reliability is often quantified using metrics such as mean time between failures (MTBF) and mean time to repair (MTTR), which help developers understand and improve the robustness of their software.

Usability is a key factor that determines how easily and effectively users can interact with the software. High usability ensures that users can accomplish their tasks efficiently, with minimal learning curve and effort. This involves intuitive user interfaces, clear and concise documentation, and responsive design that caters to the needs of diverse user groups. Usability testing and user feedback play vital roles in refining the user experience and enhancing overall satisfaction.

Efficiency in software quality pertains to the optimal use of resources, including processing power, memory, and bandwidth. Efficient software performs its tasks quickly and consumes minimal resources, leading to faster response times and reduced operational costs. Performance testing and optimization techniques are employed to identify bottlenecks and improve the software's efficiency.

Maintainability is the ease with which software can be modified to correct defects, improve performance, or adapt to changing requirements. Maintainable software is characterized by clear and well-documented code, modular design, and adherence to coding standards. It allows developers to make changes swiftly and with minimal risk of introducing new defects. Maintainability is crucial for the long-term sustainability of software, as it ensures that the software can evolve and remain relevant over time.

Portability is the software's ability to operate across different environments and platforms with minimal modification. Portable software can be easily transferred and adapted to various hardware and software configurations, enhancing its flexibility and reducing the costs associated with platform-specific development. Ensuring portability involves writing platform-independent code and using standard libraries and frameworks.

In conclusion, software quality is a comprehensive measure of the value and performance of software systems. It encompasses multiple attributes, including functionality, reliability, usability, efficiency, maintainability, and portability. Achieving high software quality is essential for delivering software that meets user expectations, operates reliably and efficiently, and can be easily maintained and adapted over time. Through rigorous testing, user-centered design, and adherence to best practices, developers can ensure that their software products achieve the highest standards of quality, ultimately leading to greater user satisfaction and success in the marketplace.

2. Methodologies for Software Quality Prediction

2.1 Statistical Models

Statistical models, such as regression analysis, have been widely used for software quality prediction. These models rely on historical data to identify relationships between code metrics and quality attributes. Linear regression, logistic regression, and discriminant analysis are common statistical techniques used in this context.

2.2 Machine Learning Models

Machine learning has revolutionized the field of software quality prediction. Techniques such as decision trees, support vector machines, neural networks, and ensemble methods have been applied to predict various quality attributes. Machine learning models can handle complex relationships and interactions between variables, making them suitable for predicting non-linear and non-monotonic quality attributes.

2.3 Hybrid Models

Hybrid models combine statistical and machine learning approaches to leverage the strengths of both. For example, a hybrid model might use statistical methods to preprocess data and machine learning to build the prediction model. These models aim to improve prediction accuracy and reduce the limitations of individual approaches.

3. Challenges in Software Quality Prediction

3.1 Data Quality and Availability

The accuracy of software quality prediction heavily depends on the quality and quantity of available data. In practice, obtaining reliable and comprehensive datasets is challenging due to the proprietary nature of software development data and the lack of standardized data collection processes.

3.2 Model Generalization

Building models that generalize well across different software projects and domains is a significant challenge. Software projects vary greatly in terms of size, complexity, and development practices, which can affect the applicability of prediction models.

3.3 Evolving Software Development Practices

The rapid evolution of software development practices, such as the adoption of agile methodologies and DevOps, poses new challenges for software quality prediction. Prediction models must adapt to these changes to remain relevant.

4. Opportunities and Future Directions

4.1 Big Data and Analytics

The advent of big data analytics presents new opportunities for software quality prediction. By leveraging large volumes of data from various sources, such as version control systems, bug tracking systems, and continuous integration servers, more accurate and comprehensive prediction models can be developed.

4.2 Deep Learning

Deep learning, a subset of machine learning, has shown promise in various domains due to its ability to learn complex patterns. Applying deep learning to software quality prediction could lead to more sophisticated models that can capture the intricate relationships between code characteristics and quality attributes.

4.3 Transfer Learning

Transfer learning, which involves applying knowledge gained from one problem to a different but related problem, could be beneficial in software quality prediction. By transferring knowledge across projects and domains, prediction models could become more robust and generalizable.

5. Related Work:

In [1], Ai-jamimi and Hamid proposed a fluffy rationale based SDP model. The presentation of this rationale based forecast model has been checked by genuine programming projects information. They track down this model as the best method for acquiring prevailing arrangement of measurements. This thus make fluffy rationale based model more legitimate and good when contrasted with different models. Result showed that utilizing all product measurements gives the most minimal exactness and less fulfillment as contrasted and the other arrangement of measurements. The applicable arrangement of measurements gives better outcome that is measurements gotten after expulsion of excess measurements.

In [2], Koroglu et al. utilized seven old renditions of programming and their extra component to track down the deformities of current forms. They analyzed a few SDP process that is Naïve Bayes, choice tree, and irregular backwoods and observes the arbitrary woods has the most noteworthy prescient power when contrasted with different models. This multitude of models are contrasted and the AUC esteem that is region under bend. They observe that irregular timberland has the most elevated AUC esteem.

In [3], Sharmin proposed an original strategy of quality determination that is choice of trait with log sifting (SAL). They utilized the log sifting to preprocess the information. At long last, reaches the resolution that this strategy gives the more exactness of SDP when contrasted with different procedures. This technique is applied on a few broadly utilized openly accessible datasets

In [4], Sethi and Gagandeep track down that the fake brain organization (ANN) gives the better outcome when contrasted with fluffy based rationale model. ANN gives the more precise worth. It tends to be utilized in half breed way to deal with an enormous dataset. These model is investigated with the mean size of relative blunder (MMRE) and adjusted mean greatness of relative mistake (BMMRE).

In [5], Suffian involved the measurements to observe the exhibition of various models that is relapse model with different models. They observe that relapse investigation is

generally exact when contrasted with different models. They utilized the p-worth of 0.05 as the edge for the choice of qualities of programming.
In [6], Ami et al. proposed an original methodology of property choice technique for development of viable deformity expectation model. This approach observes the characteristics with high exactness by working out the complete load of each property and arranging each quality in light of all out weight. They utilized the one classifier that is Naïve Bayes in their review to develop the SDP model.
In [7], Can et al. presented a clever methodology for programming deformity forecast PSO and SVM called as P-SVM model and saw that P-SVM has more precision than BP brain organization, SVM Model and GA-SVM model. They found this model as generally powerful. The dataset utilized is just JM1 for proposing the clever methodology of P-SVM.
In [8], Jiarpakdee finds in the wake of examining 101 accessible datasets that 10-67% of measurements of these datasets are excess. Additionally, it has been seen that end of excess measurements prior to building the SDP model is vital. It works on the presentation of SDP model.
In [9], Wang et al. seen that multivariant Gauss Naïve Bayes has best execution when contrasted with all sort of classifiers. It is best imperfection forecast model. They additionally explore different avenues regarding J48 to track down the presentation of multivariant Gauss Naïve Bayes. They observed that MVGNB is best in anticipating the deformities at a beginning phase of programming improvement.
In [10], Liu et al. proposed a SDP model for that help arranged programming. They observe the SDP model in view of the current model, QDPSOMO. It gives better administration of value to programming that relies upon EXPERT COCOMO. It is shaped by the blend of imperfection expectation, estimation and the board.
In [11], Kakkar and Sarika Jain closed from their examination work that cross breed model of classifier or the mix of at least one classifier generally gives the preferable outcome over any single classifier. The half breed approach of choice of quality gives more exactness. It likewise assists us with dissecting the effect of property choice and preprocessing of information on various SDP models. Execution of five classifiers has been thought about, i.e., IBk, KStar, LWL, Random backwoods, and Random tree. It has been seen that LWL gave the exactness of 92.23% and has best execution.
In [12], Verma and Kumar investigated the numerous relapse in their exploration work. They track down the effect of bunching on deformity forecast. Three bunches are shaped. Result has shown that forecast model shaped in the wake of grouping showed preferred outcome rather over applying expectation model on entire programming project.
In [13], Yang et al. proposed an original methodology that is figuring out how to-rank (LTR) approach for the development of SDP model. This approach assists with finding the test assets all the more really by observing which module of programming have more deformities. They observed that figuring out how to rank methodology gives better expectation exactness when contrasted with direct model utilizing LS. Be that as it may, LTR now and again isn't giving as better outcome as given by Random Forest. LTR isn't performing better in all cases.
In [14], Sawadpong and Allen utilize an uncommon dealing with for execution of SDP model. They proposed special case based programming measurements. It depends on the underlying ascribes of exemption dealing with call diagrams. They arrived at the resolution that assuming SDP model that is relies upon uncommon based measurements gives more outcome when contrasted with customary expectation model. They utilized the product vaults that have mined information and imperfection reports for their exploration.
In [15], Shuai et al. executed Genetic calculation with SVM (GA-CSSVM) on NASA datasets. They inferred that GA-CSSVM performed better when contrasted with increments typical SVM.
In [16], Gabriel Kofi Armah et al. performed Multilevel preprocessing by choosing the properties two times and sifting example threefold. Four K-NN classifier's preprocessing that is KNN-LWL, KStar, IBK, and IB1 results were dissected and contrasted and irregular tree, arbitrary timberland, and non-settled summed up classifier. Four execution boundary that is exactness, review, Area under bend (AUC) and accuracy are utilized to analyze them. Results showed that presentation of Random Forest expanded by performing twofold preprocessing.
In [17], Lo et al. consolidated SVM and Auto Regression Integrated Moving Average (ARIMA) for SDP. They dissected that exhibition of cross breed model is better when contrasted with customary forecast model and diminishes blunder rate.
In [18], Oral et al. performed SDP by joining three grouping procedures that is NB, casting a ballot highlight stretch and MLP utilizing five datasets. He reasoned that mix of these classifiers gives better execution to SDP models particularly for inserted framework.
In [19], Singh et al. broke down the exhibition of various mining methods that is Logistic Regression, irregular backwoods, C4.5, Association Rule Mining, Naïve Bayes, ANN, SVM, hereditary calculation and Fuzzy Programming. They presumed that Data Mining strategies are extremely useful for eliminating minor imperfections.
In [20], Challagulla et al. looked at 13 AI techniques. They track down that NB, brain organization, and Instance-based learning performed better compared to other when contrasted with any remaining strategies.
While many investigations in the product deformity forecast independently report the similar presentation of the displaying methods utilized, there is no solid agreement on which performs best when the examinations are checked person out.
Bibi et al. (Bibi, Tsoumakas, Stamelos, and Vlahavas, 2008) have detailed that Regression by means of Classification (RvC) functions admirably. Lobby et al. featured that reviews utilizing Support Vector Machine (SVM) perform less well. These might be performing roar assumption as they require boundary enhancement for the best exhibition (T. Corridor et al., 2012).
C4.5 appears to perform roar assumption in the event that they incorporate imbalanced class dispersion of datasets, as the calculation
is by all accounts delicate to this (Arisholm, Briand, and Fuglerud, 2007) (Arisholm, Briand, and Johannessen, 2010). Credulous Bayes (NB) and Logistic Regression (LR) appear

to be the techniques utilized in models that performs moderately well in the field of programming deformity expectation (Menzies et al., 2007) (Song et al., 2011). NB is a surely known calculation and regularly being used. Concentrates on utilizing Random Forests (RF) didn't proceed as well true to form (T. Lobby et al., 2012). In any case, many investigations utilizing the NASA dataset utilize RF and report great performanc (Lessmann et al., 2008).
A few investigations on programming imperfection forecast showed that Neural Network (NN) has a decent precision as a classifier
(Lessmann et al., 2008) (Benaddy and Wakrim 2012) (Quah, Mie, Thwin, and Quah, 2003) (T M Khoshgoftaar, Allen, Hudepohl, and Aud, 1997). NN has been demonstrated to be more sufficient for the issue on the muddled and nonlinear connection between programming measurements and deformity inclination of programming modules (Zheng 2010). Nonetheless, the practicability of NN is restricted because of trouble in choosing suitable boundaries of organization engineering, including number of stowed away neuron, learning rate, force and preparing cycles (Lessmann et al., 2008).
In any case, models appear to have performed best where the right method has been chosen for the right arrangement of information. No specific classifiers that plays out awesome for all the datasets (Challagulla, Bastani, and Paul, 2005) (Song et al., 2011). Subsequently, the examinations and benchmarking aftereffects of deformity expectation utilizing AI classifiers demonstrate that the unfortunate precision level is predominant (Sandhu, Kumar, and Singh, 2007) (Lessmann et al., 2008), huge execution contrasts couldn't be distinguished (Lessmann et al., 2008) and no specific classifiers play out awesome for all the datasets (Challagulla, Bastani, and Paul, 2005) (Song et al., 2011).
Karpagavadivu.K, et.al. (2012)[21] dissected the presentation of different strategies utilized in programming issue forecast. And furthermore depicted a few calculations and its purposes. They observed that the point of the shortcoming inclined module expectation utilizing information mining is to work on the nature of programming improvement process. By utilizing this procedure, programming administrator really apportion assets. The general mistake paces of all strategies are looked at and the benefits of all techniques were broke down. Ahmet Okutan and Olcay Taner Yıldız, (2013)[22] proposed another bit technique to anticipate the quantity of imperfections in the product modules (classes or documents). The proposed technique depends on a pre-processed piece lattice which depends on the likenesses among the modules of the product framework. Novel bit technique with existing bits in the writing (straight and RBF bits) has been analyzed and show that it accomplishes tantamount outcomes. Besides, the proposed deformity expectation strategy is likewise similar with some current well known imperfection forecast techniques in the writing for example straight relapse and IBK. It was seen that before test stage or support, designers can utilize the proposed technique to effortlessly anticipate the most blemished modules in the product framework and spotlight on them essentially as opposed to testing every single module in the framework. This can diminish the testing exertion and the absolute venture cost consequently. Yajnaseni Dash, Sanjay Kumar Dubey, (2012)[23] studied different examination strategies for the expectation of OO metric utilizing brain network procedures. This procedure was viewed as the most appropriate for expectation in the event of article situated measurements. Brain network utilized least estimation function when contrasted with other computerized reasoning procedures. It has better portrayal capacity and is fit for performimg muddled capacities. Ms. Puneet Jai Kaur, Ms. Pallavi, (2013)[24] involved various information digging procedures for programming mistake forecast, similar to affiliation mining, arrangement and grouping strategies. This has helped the computer programmers in growing better models. In the event that where deformity marks are absent, solo strategies can be utilized for model turn of events.
Xiaoxing Yang, et.al. (2014)[25] Used the position execution advancement method for programming determining model turn of events. For this position to it was utilized to learn approach. The model was created on past work and was subsequently read up for working on the presentation of the model. . The work incorporates two perspectives: one is an original use of the figuring out how to-rank way to deal with genuine informational indexes for programming imperfection expectation, and the other is a complete assessment and examination of the figuring out how to-rank technique against different calculations that have been utilized for foreseeing the request for programming modules as indicated by the anticipated number of deformities. This study shows that the impact of advancement of the model exhibition utilizing rank to learning approach really further develop the expectation exactness.

6. Conclusion

Software quality prediction is an essential component of modern software engineering that can lead to significant improvements in software quality and development efficiency. While there have been significant advancements in this field, challenges remain, particularly in data availability, model generalization, and adapting to evolving software development practices. Future research should focus on leveraging big data analytics, exploring deep learning techniques, and utilizing transfer learning to overcome these challenges and advance the state-of-the-art in software quality prediction.

References:

[1]. Ai-jamimi, H. A. (2016). Toward comprehensible software defect prediction models using fuzzy logic (pp. 127–130).
[2]. Koroglu, Y., Sen, A., Kutluay, D., Bayraktar, A., Tosun, Y., Cinar, M., & et al. (2016). Defect prediction on a legacy industrial software : A case study on software with few defects. In 2016 IEEE/ACM 4th International Workshop on Conducting Empirical Studies in Industry (CESI) (pp. 14–20).
[3]. Sharmin, S. (2015). SAL: An effective method for software defect prediction (pp. 184–189).
[4]. Sethi, T., & Gagandeep. (2016). Improved approach for software defect prediction using artificial neural networks. In 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (pp. 480–485).

[5]. Suffian, M. D. M., Ibrahim, S., Dhiauddin, M., Suffian, M. D. M., & Ibrahim, S. (2012). A prediction model for system testing defects using regression analysis. International Journal of Soft Computing and Software Engineering, 2(7), 69–78.
[6]. Mandal, P.,&Ami, A. S. (2015). Selecting best attributes for software defect prediction. In 2015 IEEE International WIE Conference on Electrical and Computer Engineering (pp. 110–113).
[7]. Can, H., Jianchun, X., Ruide, Z., Juelong, L., Qiliang, Y., & Liqiang, X. (2013). A new model for software defect prediction using Particle Swarm Optimization and support vector machine. In 2013 25th Chinese Control and Decision Conference (pp. 4106–4110).
[8]. Jiarpakdee, J., Tantithamthavorn, C., Ihara, A., & Matsumoto, K. (2011). A study of redundant metrics in defect prediction datasets (pp. 37–38).
[9]. Wang, T.,&Li,W. (2010).NaïveBayes software defect predictionmodel. IEEE, no. 2006 (pp. 0–3).
[10]. Liu, J., Xu, Z., Qiao, J., & Lin, S. (2009). A defect prediction model for software based on service oriented architecture using EXPERT COCOMO. In 2009 Chinese Control and Decision Conference (pp. 2591–2594).
[11]. Kakkar, M., & Jain, S. (2016, January). Feature selection in software defect prediction: A comparative study. In 2016 6th International Conference on Cloud System and Big Data Engineering (Confluence), (pp. 658–663).
[12]. Verma, D. K., & Kumar, S. (2015). Emperical study of defects dependency on software metrics using clustering approach (pp. 0–4).
[13]. Yang, X., Tang, K., & Yao, X. (2015). A learning-to-rank approach to software defect prediction. IEEE Transactions on Reliability, 64(1), 234–246.
[14]. Sawadpong, P., & Allen, E. B. (2016). Software defect prediction using exception handling call graphs : A case study.
[15]. Shuai, B., Li, H., Li, M., Zhang, Q., & Tang, C. (2013). Software defect prediction using dynamic support vector machine. In 2013 9th International Conference on Computational Intelligence and Security (CIS) (pp. 260–263).
[16]. Armah, G. K., Luo, G., & Qin, K. (2013). Multi_level data pre_processing for software defect prediction. In 2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII) (pp. 170–174).
[17]. Lo, J.-H. (2012). A data-driven model for software reliability prediction. In IEEE International Conference on Granular Computing.
[18]. Oral, A. D., & Bener, A. B. (2007, November). Defect prediction for embedded software. In 22nd International Symposium on Computer and Information Sciences, 2007. ISCIS 2007 (pp. 1–6). New York: IEEE.
[19]. Singh, A., & Singh, R. (2013, March). Assuring Software Quality using data mining methodology: A literature study. In 2013 International Conference on Information Systems and Computer Networks (ISCON) (pp. 108–113). New York: IEEE.
[20]. Challagulla, V. U. B., Bastani, F. B., Yen, I. L., & Paul, R. A. (2008). Empirical assessment of machine learning based software defect prediction techniques. International Journal on Artificial Intelligence Tools, 17(02), 389–400.
[21] 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.
[22] Ahmet Okutan1 and Olcay Taner Yıldız, (2013), "A Novel Regression Method for Software Defect Prediction with Kernel Methods", ICPMRA 2013 - International Conference on Pattern Recognition Applications and Methods, pp 216-221.
[23] 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.
[24] Ms. Puneet Jai Kaur, Ms.Pallavi, (2013), "Data Mining Techniques for Software Defect Prediction", International Journal of Software and Web Sciences (IJSWS),International Journal of Software and Web Sciences 3(1), pp. 54-57.
[25] Xiaoxing Yang, et.al. (2014), IEEE TRANSACTIONS ON RELIABILITY, This article has been accepted for inclusion in a future issue of this journal.