A Review of Machine Learning Techniques for Software Quality Prediction

Sanjeev K. Cowlessur, Saumendra Pattnaik and Binod Kumar Pattanayak

Abstract Successful implementation of a software product entirely depends on the quality of the software developed. However, prediction of the quality of a software product prior to its implementation in real-world applications presents significant challenges to the software developer during the process of development. A limited spectrum of research in this area has been reported in the literature as of today. Most of the researchers have concentrated their research work on software quality prediction using various machine learning techniques. Another aspect pertaining to software quality prediction is that the prediction must be achieved in the earlier stages of software development life cycle in order to reduce the amount of effort required by the developer in course of the development of a software product. In this paper, we carry out a comprehensive review of machine learning techniques which have been used to predict software quality.

Keywords Software development · Software quality · Machine learning · Quality prediction · Quality assessment

S. K. Cowlessur (✉)
Department of Software Engineering, Université des Mascareignes, Beau Bassin-Rose Hill, Mauritius
e-mail: scowlessur@udm.ac.mu

S. Pattnaik · B. K. Pattanayak
Department of Computer Science and Engineering, Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar, India
e-mail: saumendrapattnaik@soa.ac.in

B. K. Pattanayak
e-mail: binodpattanayak@soa.ac.in

B. Pati et al. (eds.), *Advanced Computing and Intelligent Engineering*, Advances in Intelligent Systems and Computing 1089,
https://doi.org/10.1007/978-981-15-1483-8_45

1 Introduction

The quality of a software product can be defined as the measure of performance of a system on which the software is implemented in terms of execution time, memory capacity utilized and probability of errors, etc. In addition to this, the amount of effort contributed by the software developer also represents a key factor while assessing the quality of a software product. The quality of a software product can be considered to be internal as well as external. The internal quality of a software can be assessed in course of development during software development life cycle (SDLC); whereas, the external quality can be measured during its implementation and can be assessed with respect to its level of functionality. The external quality also depends upon its internal quality. In order to assess the external quality of a software product, quality models can be devised that represent a function of the internal quality attributes. In order to achieve this, first of all the internal attributes must be identified and then the relationship existing between the internal and external quality attributes must be identified. A number of software quality prediction models have been proposed by various authors. However, the machine learning approach to devising such a model appears to be more popular and more effective as claimed by the authors. We have been motivated by this aspect to carry out a review of the machine learning approaches for software quality prediction models.

In this work, we mainly focus on the following machine learning techniques: evolutionary algorithms, artificial neural networks (ANN), Bayesian networks (BN), fuzzy logic (FL), decision tree (DT), support vector machine (SVM) and case-based reasoning (CBR). We provide a detailed description of different models used for the prediction of software quality as proposed by various authors in each of the techniques mentioned above. The rest of the paper is organized as follows. Section 2 includes the discussion of various software quality models proposed by different authors using various machine learning techniques mentioned above. Section 3 concludes the paper along with probable future extensions.

2 Machine Learning Approaches to Software Quality Prediction

In this section, we discuss different machine learning approaches used by various authors for prediction of software quality.

2.1 Evolutionary Computing Technique-Based Approaches

In this section, we bring forth a few approaches as proposed by different authors to predict software quality making use of evolutionary computing techniques such as genetic algorithm (GA), ant colony optimization (ACO) and several other techniques as well.

2.1.1 Software Faulty Module Identification Using GA

In practice, during implementation of a software product, often errors are encountered for the reason that the software may consist of faulty modules which need to be identified, and such a task is tedious for a software professional. Moreover, this problem intensifies in case of problem-solving software. GA, being more of a problem-solving algorithm, can presumably be useful for identification of such faulty modules in a software. A GA-based approach for faulty module identification of open software was proposed by the authors in [1], wherein the following sequence of steps needs to be followed:

Step 1: Data collection of data from source code;
Step 2: Evaluation of collected data in terms of the metrics such as lines of code (LOC), depth of inheritance (DOI), number of children (NOC), lack of cohesion (LOCM), coupling between objects (CBO), number of public methods (NPM), response for a class (RFC) and weighted methods per class (WMC);
Step 3: Identification of appropriate metrics for fault prediction by virtue of filtration of data refined in step 2;
Step 4: Processing of data obtained from step 3 using GA;
Step 5: Assessment of performance via confusion metrics and analysis of prediction.

Here, fault-prone classes along with metrics are identified using GA. The authors claim a high accuracy of fault prediction using GA as compared to other approaches even though it principally relies on evolutionary computation techniques.

2.1.2 Fault Table for Fault Identification Using GA

In this approach, the faults in the current version of a software product can be comfortably anticipated with reference to the history of fault occurrence pertaining to the previous version of the same software that is maintained in a fault table. In the absence of such a table, the classification of faults characteristic to that particular software can be useful. A fault classification technique is proposed in [2] where the authors use GA for fault classification using a supervised learning method. Here, authors identify the most appropriate attributes of the software in consideration and then classify the attributes as faulty and non-faulty ones using GA in the context of an example following which as claimed by the authors, the faulty attributes need revision.

2.1.3 Software Quality Prediction Using ACO

It becomes difficult to predict the quality of a software during the early stages of the SDLC due to the fact that many of the software attributes at this point in time remain immeasurable. However, this can be derived from the attributes that are measurable at that point in time. A quality prediction model can serve this purpose. An ACO-based approach is proposed by authors in [3] where they assume adaptation of an existing model for prediction of software quality that is designed for one data set to a new data set. The authors conduct their experiments on an object-oriented system thereby claiming that it can also be used for various other software systems too. Authors conclude that such an approach can provide much better results as compared to other machine learning approaches.

2.1.4 Software Behaviour Estimation Using SPS

There are several software tools available that can successfully be used for estimating the behaviour of a software product. Software project simulator (SPS) belongs to such class of tools which can necessarily estimate the possible outcomes from the decisions taken by the developer and thus the behaviour of the software as a whole. The model proposed in [4] aims at running an evolutionary algorithm on SPS that is capable of yielding in the necessary decision rules used by a developer in course of development of a software product which can presumably facilitate the estimation of software behaviour during its implementation. As per this approach, SPS is expected to generate the database for a specific project that would represent the input to an evolutionary algorithm which subsequently provides the management rules to be taken up by the software developer during development. A management rule, thus, can predict a probable outcome from every action that is executed by the developer. It can help the developer to pick one or other action as per requirement, and subsequently the software behaviour can be usefully estimated further.

2.2 Artificial Neural Network (ANN)-Based Approaches

In this section, we discuss various artificial neural network-based approaches to software quality prediction.

2.2.1 An ANN Approach to Software Reliability Assessment

The quality attributes of a software product can be useful in verifying various functionality-related aspects of it that are mostly related to the internal organization of the system. These attributes can necessarily reveal the reliability of the software system. Artificial neural network (ANN) approach can be very useful in this regard. A software reliability growth model is proposed by the authors in [5] which relies on a neuro-fuzzy inference system that can provide a significant measure of the reliability of the software.

2.2.2 ANN-Based Software Reliability Prediction Model

Conventional models used for prediction of software quality mostly function on a set of assumptions made relating to the development environment, and due to this reason, such models may not be suitable for sufficiently complex applications. To overcome such an issue, a reliability prediction model needs to be free from any assumptions, and such a model can be viable using ANN approach as suggested in [6]. This model takes as input the history of failures of the software and can successfully predict the probable future failures, and most importantly, such a model can necessarily scale with the complexity of the software.

2.2.3 ANN-Based Approach for Software Fault Prediction Models

As claimed by the authors in [7], ANN model can be useful in determining the occurrence of the anticipated number of faults in a software product during its implementation, and the authors justify the same having proposed an ANN model for software fault prediction with reference to a hypothesis that suggests that the distribution of faults in a software product mostly depends on its complexity. Using an experiment, the authors conclude that it performs better than other existing software fault prediction models.

2.3 Bayesian Network (BN) for Software Quality Prediction

In this section, we bring a review of various Bayesian network (BN)-based software quality prediction models.

2.3.1 Activity-Based BN Models

In [8], the authors proposed an activity-based quality model (ABQM) for software quality prediction as well as assessment using BN approach. Here, authors propose to split a more complex concept into simpler definitions, and for each definition, a node is created in BN. Further, BN is enriched with quality-related quantitative information that can predict the software quality with much perfection.

2.3.2 Integration Software Quality Prediction Model

Most of the prediction models for software quality are oriented towards the prediction of only a single quality feature. But in practice, any software is associated with a set of quality features such as functionality, efficiency, reliability, reusability and many others. It is necessary that a prediction model be capable of predicting the behaviour of multiple features. In [9], the authors develop a BN model that is aimed at predicting two quality features, namely reliability and usability. Here, the authors split each feature into a set of sub-features, and each sub-feature is assigned a node in the network. As claimed by the authors, this model can be scaled to more quality features, and thus, more accurate quality prediction can be achieved using such a model.

2.3.3 Quality Prediction in Extreme Programming

Extreme programming (XP) is an iterative approach to software development which comprises components called user stories where each user story is equivalent to software requirement specification (SRS) document in software development. However, quality prediction in XP processes is extremely difficult. A BN-based quality prediction model for XP processes is proposed in [10] that is capable of predicting the quality during the planning stage well before the development starts. It can also predict the time when the project will be completed.

2.3.4 Quality Prediction of Embedded Software

A Bayesian belief network (BBN) is proposed by in [11] that focuses on the quality prediction of embedded software. This model mostly relies on various relationships that exist between different processes involved in development of embedded software. Authors claim that BBN is much more efficient in the quality prediction of embedded software.

2.4 Fuzzy Logic-Based Prediction Models

In this section, fuzzy logic (FL)-based software quality prediction models proposed by various authors are detailed.

2.4.1 Triangular Fuzzy Membership Function for Quality Assessment

Prior to the assessment of quality of a software product, a thorough investigation of software metrics must be done. A FL-based quality assessment procedure is proposed by authors in [12]. Here, the authors take into consideration two metrics for quality assessment, namely error rate and rate of inspection. The authors, in their approach, have used triangular fuzzy membership function for the representation of these metrics and applied them to a set of fuzzy inference rules. They have conducted experiments over a number of modules and successfully identified the error-prone modules.

2.4.2 Transparent Model for Maintainability Prediction

Most of the software quality assessment models do not take into consideration two crucial quality aspects: imprecise linguistic knowledge regarding the software obtained from the experts and precise quantitative information obtained from historical data of implementation of the software. A FL-based transparent software quality assessment model has been developed by the authors in [13] which incorporates these two forms of knowledge in order to predict software maintainability. Here, Mamdani fuzzy inference has been implemented by the authors that demonstrates significantly better performance for maintainability prediction as compared to other machine learning techniques used for this purpose.

2.4.3 Integrated Software Quality Assessment

In order to develop a software quality prediction model, it is essential to take into account the most relevant quality attributes and their accurate quantification. However, it is associated with two major aspects: choice of the source of the relevant attributes and uncertainty associated with their accurate quantification. A fuzzy multi-criteria-based integrated software quality evaluation model is proposed in [14] with the quality attributes having been picked from ISO/IEC 9126 standard. Results justify the efficiency of this model in quality prediction.

2.4.4 Evaluation of Software Similarity

Quality evaluation between two different software products can be achieved from the level of similarity existing between them in the context of the amount of effort consumed in the development process. This aspect can be rarely found in the literature pertaining to software quality prediction. This problem of similarity evaluation has been addressed by the authors in [15], wherein the level of similarity between two software projects is attributed as the distance in quality between the two projects. Nevertheless, the similarity cannot be expressed in numerical values for which the authors have used FL to assign linguistic values to the level of similarity between two software projects which can further be defuzzified to obtain a numerical assessment.

2.4.5 Neuro-Fuzzy Model for Quality Prediction Model

A hybrid neuro-fuzzy model for software quality prediction has been proposed by authors in [16] where the authors use the software attributes and the sub-attributes which are analysed by a fuzzy logic neural network (FLNN) system developed by them. A set of fuzzy If–Then rules are used for this purpose. This model can be useful during the process of software development.

2.5 Decision Tree (DT) Approach for Quality Prediction Model

In this section, we detail the spectrum of research work carried out by different authors relating to software quality prediction making use of decision tree approach.

2.5.1 Quality Classification Using DT

By virtue of prediction of quality of a software product prior to testing can significantly reduce the quantum of work of the developer during software development. DT approach-based model for classification of software quality that relies on most relevant software metrics is capable of identifying the faulty modules in a software product. Researchers attribute such models to white box quality assessment models. DT approach-based algorithm SPRINT is used by the authors in [17] for the classification of software quality and obtains successfully the relevant classification trees that can contribute to the classification of software quality. This approach is free from any memory bottleneck that can be found in other algorithms implemented for this purpose. SPRINT algorithm is an extended version of DT algorithm CART that is also used for classification of software quality.

2.5.2 DT-Based Quality Models Using Principal Component Analysis (PCA)

Quality prediction models based on classification trees make use of metrics related to the software product, software process as well as execution of it for identification of faulty modules in a software. As observed, these models are found to be more robust as compared to other models. However, the accuracy of identification in case of such models may present a challenge. The authors in [18] have conducted a case study of four different versions of large communication systems making use of principal component analysis (PCA) DT-based models which can be further improved by virtue of transformation of quality prediction algorithms.

2.5.3 DT-Based Quality Assessment Models for Fault Prediction

Identification of high-risk modules in a software product at early stages of development can make it feasible for improvement of anticipated faulty modules that can presumably enhance the reliability of the software during its implementation. Regression tree-based models can be significantly more effective in prediction of software reliability than any other models. A regression tree approach for quality prediction models has been addressed by authors in [19], wherein a case study is conducted with large telecommunication systems with lines of code of more than 13 million making use of regression tree algorithms like CART-LS and S-PLUS, and the authors conclude that identification of faulty modules in a software can be achieved at ease using this approach. The fault prediction accuracy has been assessed in terms of the metrics such as absolute, average and relative errors.

2.6 Support Vector Machine (SVM)-Based Prediction Models

In this section, we discuss different models for software quality prediction using support vector machine (SVM) approach.

2.6.1 SVM for Prediction of Software Defects

Defects in software products can be removed during testing of the software. However, if such defects can be identified at earlier stages prior to testing during software development, then significant amount of time and effort can be saved. SVM approach can be used to identify such defects well in advance. SVM approach is used by authors in [20] for defect prediction, and the authors claim to have achieved a high accuracy in prediction. Here, authors use MATLAB as the interface and implement SVM on the ANT-1.7 data set for identification of the defects.

2.6.2 SVM Classifier for Code Fault Prediction

Machine learning classifiers and SVM classifiers, in particular, have been proved to be significantly useful in the prediction of fault in the code of a software. In order to achieve this prediction, the classifier needs to be trained in the historical data. However, if the number of features is very large, then the accuracy may be reduced. A feature selection technique has been proposed by authors in [21] which can be used for fault prediction in software code. Here, authors use Naive Bayes and F-measure metric for improvement of accuracy of prediction.

2.6.3 Prediction of Defects in Software Using SVM

Identification and removal of software defects are a time-consuming process, and if a software project is not planned properly, then the process of identification and removal of defects consumes significantly more time than the process of code development. A SVM-based technique proposed by authors in [22] for software defect prediction uses a data mining approach for the identification of software defects on the basis of existing software metrics. Authors claim significant improvement of software quality using their technique that subsequently reduces the development cost significantly.

2.6.4 Empirical Approach to Defect Detection Using SVM

An empirical approach to software defect prediction is proposed by authors in [23] that uses SVM approach. Here, authors have chosen code smells as the feature for the prediction. First of all, a smell prediction model is developed using SVM technique that is subsequently used for software defect prediction. The authors revealed in their paper that this technique was used in the development of subsequent release of the Eclipse software and claims high accuracy in prediction.

2.7 Case-Based Reasoning for Quality Prediction Model

In this section, various techniques for software quality prediction using a machine learning technique called case-based reasoning (CBR) are detailed.

2.7.1 CBR for Software Quality Estimation

Authors in [24] propose a CBR technique for software quality estimation procedure that can be conducted by human experts. This approach mostly relies on the similarity that exists between projects in the past which is achieved taking into account some key quality attributes of the software. Various similarity techniques are conducted by the authors in order to find out the optimal one that resulted in much higher accuracy as well as reliability in the estimation of software quality.

2.7.2 CBR for Prediction of Faulty Modules of Software

If the faulty modules of a software can be identified in advance, the quality of the software can be presumably improved. Authors in [25] propose a CBR-based technique for prediction of fault-prone modules within a software during the process of development. Here, authors refer to the hypothesis that a module is supposed to be fault-prone if a module developed earlier with the same functionality was found to be fault-prone as well.

2.7.3 CBR for Estimation of Development Effort

If the amount of human effort contributed in software development can be assessed, the quality of the software can be accordingly improved that consequently reduces the cost of the software. Authors in [26] operate on a CBR approach for effort estimation during development with respect to student programs. Here, the CBR model proposed by authors validates the student data and mainly concentrates on the imprecision and uncertainty related to the chosen attributes from student data. This model is mainly intended at reducing the cost of effort involved in the development of software.

2.7.4 Fault Prediction Using CBR

Most of the prediction models of software quality that use CBR approach focus on the similarity distance between developed software modules. The authors in [27] have proposed a quality prediction model based on CBR approach taking into consideration the similarity distance between software modules, wherein they combine similarity function with a solution algorithm for the exploration of the fault prediction procedure. Here, authors have conducted a wide spectrum of case studies with large telecommunication legacy systems and claim that using similarity function significantly improves fault prediction in a software product.

3 Conclusion and Future Work

In this review paper, we have conducted a thorough investigation through the literature available up to date on the machine learning techniques applied to software quality prediction by a wide spectrum of authors. Different models for software quality prediction have been proposed using machine learning techniques such as ANN, BN, FL, DT, SVM and CBR and are detailed in this paper. In most cases, the authors conclude that machine learning techniques prove to be significantly more efficient in software quality prediction in comparison with other techniques. However, still there exists ample scope for implementation of some innovative approaches pertaining to machine learning techniques to be tested for prediction of the quality of a software product. The contents of this paper can serve a useful guide to the researchers in the field of software engineering in order to carry out further research work in the development of some novel software quality prediction models.

References

1. Puri, A., Singh, H.: Genetic algorithm based approach for finding faulty modules in open source software systems. Int. J. Comput. Sci. Eng. Surv. (IJCSES) **5**(3), 29–40 (2014)
2. Adline, A., Ramachandran, M.: Predicting the software quality using the method of genetic algorithm. Int. J. Adv. Res. Electr. Electron. Instrum. Eng. **3**(2), 390–398 (2014)
3. Azar, D., Vybibhal, J.: An ant colony optimization algorithm to improve software quality prediction models: case of class stability. Inf. Softw. Technol. **53**(4), 388–393 (2011)
4. Pattnaik, S., Pattanayak, B.K.: A survey on machine learning techniques used for software quality prediction. Int. J. Reasoning-Based Intell. Syst. **12**(1/2), 3–14 (2016)
5. Kapur, P.K., Khatri, S.K., Goswami, D.N.: A generalized dynamic integration software reliability growth model based on neural network approach. In: Proceedings of International Conference on Reliability, Safety and Quality Engineering, pp. 831–838 (2008)
6. Karunanithi, N., Whitley, D., Malaiya, Y.K.: Using neural networks in reliability prediction. Softw. IEEE **9**(4), 53–59 (1992)
7. Khoshgoftaar, T.M., Pandya, A.S., More, H.B.: A neural network approach for predicting software development faults. In: Proceedings of Software Reliability Engineering in 3rd International Symposium on IEEE, pp. 83–89 (1992)
8. Wagner, S.: A bayesian network approach to assess and predict software quality using activity-based quality model. Inf. Softw. Technol. **52**(11), 1230–1241 (2010)
9. Radinski, L.: A conceptual bayesian net model for integrated software quality prediction. Ann. UMCA Informatica **11**(4), 49–60 (2011)
10. Abouela, M., Benedicenti, L.: A bayesian network based XP process modelling. Int. J. Softw. Eng. Appl. (IJSEA) **1**(3), 1–15 (2010)
11. Amasaki, S., Takagi, Y., Mizuno, O., Kikuno, T.: Constructing a bayesian belief network to predict final quality in embedded system development. IEICE Trans. Inf. Syst. **8**(6), 1134–1141 (2005)
12. Mittal, H., Bhatia, P., Goswami, P.: Software quality assessment based on Fuzzy logic technique. Int. J. Soft Comput. Appl. (IJSCA) **1**(3), 105–112 (2008)
13. Ahmed, M.A., AL-Jamini, H.A.: Machine learning approaches for predicting software maintainability: a Fuzzy based transparent model. IET Softw. **7**(6), 317–326 (2013)
14. Challa, J.S., Paul, A., Dada, Y., Nerella, V., Stivastava, P.R., Singh, A.P.: Integrated software quality evaluation: a Fuzzy multi-criteria approach. J. Inf. Process Syst. (JIPS) **7**(3), 473–518 (2011)
15. Idri, A., Abra, A.: A fuzzy logic based measures for software project similarity: validation and possible improvements. In: Proceedings of 7th International Symposium on Software Metrics, pp. 85–96. IEEE, England, UK (2001)
16. Pattnaik, S., Pattanayak, B.K., Patnail, S.: Prediction of software quality using neuro-fuzzy model. Int. J. Intell. Enterp. (IJIE) **5**(3), 292–307 (2018)
17. Khoshgoftaar, T.M., Seliya, N.: Software quality classification modelling using SPRINT decision tree algorithm. Int. J. Artif Intell. Tools (IJAIT) **12**(3), 207–225 (2002)
18. Khoshgoftaar, T.M., Shah, R.M., Allen, E.B.: Improved tree-based models for software quality with principal component analysis. In: Proceedings of Software Engineering Reliability Engineering, ISSRE 2000 of 11th International Symposium on IEEE, pp. 198–209 (2000)

19. Khoshgoftaar, T.M., Seliya, N.: Tree based software quality estimation models for fault prediction. In: Proceedings of Software Metrics of Eighth IEEE Symposium, pp. 203–214 (2002)
20. Gupta, R.: Software defects prediction using Support vector machine. Int. J. Comput. Sci. Softw. Eng. **1**(1), 39–48 (2015)
21. Shankar, K., Kannan, S., Jennifer, P.: Prediction of code fault using naive bayes and SVM classifiers. Middle East J. Sci. Res. (MEJSR) **20**(1), 108–113 (2014)
22. Selvaraj, P.A., Thangaraj, P.: Support vector machine for software defect detection. Int. J. Eng. Technol. Res. (IJETRE) **1**(2), 68–76 (2013)
23. Reshi, J.A., Sing, S.: Predicting software defects through SVM: an empirical approach. Int. J. Sci. Res. Dev. **5**, 1835–1838 (2017)
24. Rashid, E., Patnaik, S., Bhattacherjee, V.: Software quality estimation using machine learning case-based reasoning technique. Int. J. Comput. Appl. (IJCA) **58**(14), 43–48 (2012)
25. Khoshgoftaar, T.M., Ganesan, K., Allen, E.B., Ross, F.D., Munikoti, R., Goel, N., Nandi, A.: Predicting fault-prone modules with case-based reasoning. In: Proceedings of Software Reliability Engineering, The Eighth International Symposium on IEEE, 27–35 (1997)
26. Rashid, E., Bhattacherjee, V., Patnaik, S.: The application of case-based reasoning to estimation of software development effort. Int. J. Comput. Sci. Inform. (IJCSI) **1**(3), 20–34 (2012)
27. Khoshgoftaar, T.M., Seliya, N., Sundaresh, N.: An empirical study of predicting software fault with case-based reasoning. Proc. Softw. Qual. J. Springer Sci. **14**(2), 85–111 (2006)