International Journal of Research in Engineering, Science and Management
Volume 7, Issue 4, April 2024
https://www.ijresm.com | ISSN (Online): 2581-5792

An Experimental Study Using Machine Learning Techniques for Software Quality Prediction

Chaitanya Salika[1], Purna Santhosh Narra[2], Sai Siddartha Reddy Satti[3], Anand Mylabathula[4*], Srihari Siva Shankar Barre[5], Aravind Yeluri[6]

***Abstract*: Software quality assessment is a necessary task at different phases of software development. It can be applied to project quality assurance practice design and benchmarking. Software quality was assessed using two methodologies in earlier research: multiple criteria linear programming and multiple criteria quadratic programming. The quality of C5.0, SVM, and neural mesh was also examined. These studies' accuracy is quite low. We attempted to increase the estimation accuracy in this work by utilizing pertinent information from a sizable dataset. To increase accuracy, we employed a correlation matrix and a non-selective approach. We also evaluated several new techniques that have worked well for previous forecasting assignments. Machine learning techniques including MLP Classifier, Random Forest, Decision Tree, XG Boost, and Logistic Regression and the data is subjected to Naive Bayes. forecast the quality of software and show how development parameters relate to quality. Results from the experiment indicate that machine learning algorithms are capable of accurately assessing the software quality level.**

***Keywords*: extreme gradient net, boosting, software quality, machine learning, estimation.**

1. Overview

A. Inspiration

Precise assessment techniques become necessary as software quality development cycles gain relevance. Prior methods like Extreme Gradient Decent, Multiple Criteria Key Words: evaluation, machine learning, software quality Improvement of a Linear Quadratic Machine learning and programming models yielded less-than-ideal accuracy. Modern modest data sets are not included in the process of improving accuracy leverage. Through the use of sophisticated machine learning models, correlation analysis, and feature selection, this study seeks to increase estimation accuracy. Redefining software quality evaluation is the aim.

B. Problem description

The limited accuracy of current software quality evaluation methods makes it more difficult to plan projects effectively and analyze prototypes. Earlier methods, such as different machine learning models and Multiple Criteria LP QP, have not produced the required degree of accuracy. Creating a reliable, accurate estimation model that makes good use of significant dataset properties and raises the predictive accuracy of software quality indicators is the difficult part.

C. Project objective

This research aims to increase the quality of software by utilizing significant traits found in large datasets to evaluate accuracy. The objective is to develop a complicated model by utilizing correlation matrices, selection techniques, and sophisticated machine learning algorithms including XG Boost, Random Forest, Decision Tree, logistic regression, MLP classifier, and Naive Bayes. This model should be able to forecast software quality with accuracy and show the intricate correlations between development features and quality levels.

D. Range

The project's primary goal is to measure the quality of software development by combining cutting-edge techniques with thorough data analysis. To predict the software quality, it uses a combination of machine learning algorithms, correlation matrices, and feature selection techniques. The investigation of these approaches' efficacy, accurate software quality level estimation, and comprehension of the relationship between development attributes and quality measurements are all included in the scope.

E. Overview of the Project

Project applications may have faults as a result of software development processes such as requirements analysis, definition, and others. Thus, one task that is required in each of their phases is the examination of software quality. It is applicable to the planning and development of project-based quality control procedures. Furthermore, one of the key indicators of software quality is thought to be the quantity of flaws per unit.

System performance can be used to define the quality of a software product. The program is executed based on variables like startup time, memory capacity, load capacity, error likelihood, etc. A crucial consideration in assessing the software's quality is also the developer's level of involvement. Software quality can be classified as internal or external. It is possible to assess the software's internal quality. While external

*Corresponding author: anandmylabathula1601@gmail.com

quality can be determined throughout implementation and evaluated based on its functionality level during the four software development life cycle (SDLC).

Its interior quality influences its external quality as well. Quality models that act as a function of internal quality attributes can be developed to evaluate the exterior quality of software.

Finding the internal characteristics and the connections between the internal and external traits are the initial steps in achieving this. Numerous writers have put forth various software quality prediction models. As the authors state, however, the machine learning method of creating such a model appears to be more widely used and successful. Our understanding of machine learning techniques for software quality prediction models was spurred by this fact.

2. Survey of Literature

[1] Sanjeev Cowlessur, Sumendra Pattnaik, and Binod Pattanayak. (2020). A Synopsis of Machine Learning Methods Canton Questionnaire.

The program itself is the only factor that determines whether or not an implementation is successful. However, during the development process, programmers have major hurdles in forecasting the quality of software before it is deployed in real applications. Until now, the literature has only reported a small amount of study in this field. The majority of researchers have concentrated their efforts on applying different machine learning approaches to forecast software quality.

[2] In 2020, Ankara, Turkey hosted the International Congress on Human Computer Interaction, Optimization, and Robotics Application (HORA). A. A. Ceran and O. X. Tanriover presented their "Experimental investigation of software quality\prediction using machine learning methods".

Software quality evaluation is a necessary task at different phases of software development. Planning and preparing quality practices for projects can be done with it. Two methods (Multiple Criteria Quadratic Programming and Multiple Criteria Programming) were employed in earlier research to assess the quality of software, together with the SVM network and 0 no. Quality Assessment.

[3] Saumendra Pattnaik and Binod Pattanayak (2016). an investigation on machine learning methods for software quality prediction. International Journal of Reasoning-Based Intelligent Systems.

Predicting software quality has become crucial in the present software development environment for a program's successful adoption in a real application and for enhancing its long-term usefulness. Additionally, to save effort in this procedure, careful identification of malfunctioning software modules is necessary during the software development process. Many writers have conducted in-depth studies in the area of machine learning techniques, which are thought to be the most effective methods for predicting the quality of software. In this work, I thoroughly investigate a number of machine learning approaches, including fuzzy logic, neural networks, and Bayesian models, among others, that are employed to forecast the quality of software and help analysts defend their suggested fixes.

[4] Bernstein, Joseph; Huang, Bing; Li, Xiaojun; Li, Ming; Smidts, Carol (2005). A research on how software reliability is affected by hardware malfunctions.

Reliable software is desired by all parties involved as it plays an increasingly significant role in modern life. A software malfunction in the computer's hardware is one of the reasons of software failure. Failure reporting has historically been used to look into the effects of these hardware malfunctions. The lack of historical data on injected faults and the possibility that, as a result, identified faults could not be real defects are two issues with user faulting that have been brought up.

3. Analysis of the System

A. Current Framework

Software quality evaluation has been the subject of prior research employing techniques like multiple-criteria linear programming and multiple-quadratic programming. Furthermore, methods including C5.0, SVM, and neural networks were attempted; nevertheless, their accuracy was only about 80%. These many methods attempted to evaluate the quality of software, but their inability to attain high accuracy remained restricted in this domain.

B. Drawbacks

1) Scalability and Complexity

Numerous approaches encounter difficulties when attempting to scale numerous criteria or intricate linkages that impact their computational usefulness and scalability.

2) Sensitivity and overfitting:

Various methods, inaccuracies, data sensitivity/imbalance, and noise all have an impact on the models' accuracy and dependability.

3) Transparency and interpretability:

Certain techniques, such neural networks and intricate algorithms, are not very clear in their decision-making; as a result, it can be challenging to interpret the outcomes and comprehend the model's rationale.

4) Computer specifications:

Huge processing resources are needed for some techniques, including SVM and neural networks, which restricts their use, particularly for huge datasets or situations with limited resources.

5) Processing Complexity Limitations

Accurately evaluating software quality qualities is impacted by the fact that many of these approaches still struggle to capture complicated, non-linear connections.

6) The Suggested Framework

The objective of this study is to increase assessment accuracy by utilizing a big database collection of crucial functions. We employed the correlation matrix and the selection procedure to increase accuracy. We also experimented with the most recent techniques that have worked well for other forecasting jobs. Software quality is predicted using machine learning methods such as XgBoost, Random Forest, Decision Tree, Logistic Regression, MLP Classifier, and Naive Bayes.

C. Benefits

1. Ensemble Methods (XgBoost, Random Forest): excellent application to big datasets, resistance to overfitting, and good accuracy.
2. Decision trees are suitable for smaller datasets, require less data processing, and are simple to comprehend.
3. Logistic regression: quick training, less prone to intersection, and effective binary classification.
4. MLP classification: adaptable to intricate relationships, effective to more complicated data kinds, and hyperparameters that can be adjusted.
5. Naive Bayes: Capability to learn from tiny amounts of training data, stability with little features, and resilience.

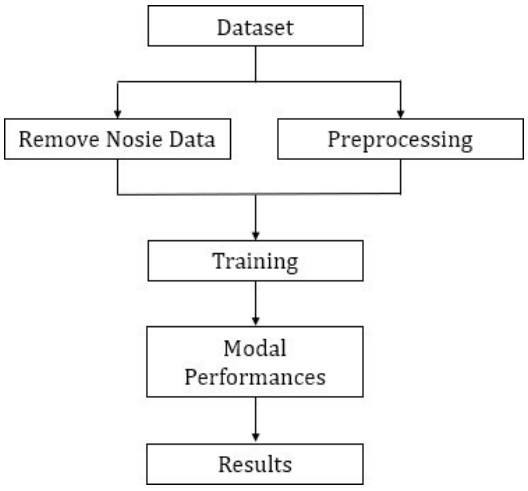

Fig. 1. System's flow

4. Needs Analyzation

A. Requirements, Both Functional and Non-Functional:

An essential procedure for assessing a system or software project's success is requirements analysis. There are two main categories of requirements: functional requirements and non-functional requirements.

Functional specifications:

Requirements that the end user considers necessary for the system to perform fundamental tasks. As stipulated in the contract, the system must have all of these features. They are expressed or defined as the following: the input provided to the system, the action taken, and the anticipated result. These are not functional needs; rather, they are essentially user- specified criteria that will be apparent in the finished product.

Functional requirements examples include:

1) Authentication of the user each time they access the system.
2) In the event of a cyberattack, system shutdown.
3) After registering for the first time in the software system, the user sent a confirmation email.

Requirements that are not functional:

These are basically quality constraints that the system has to meet in order for the project contract to be fulfilled. The degree to which these elements are used or prioritized differs depending on the project. We also refer to these as non-behavioral needs.

They mostly deal with the following problems:

- Mobility
- Safety
- Sturdiness
- Dependability
- Flexibility
- Effectiveness
- Adaptability
- Adaptability

Non-functional needs examples include:

1) Within 12 hours after such an incident, emails should be issued.
2) Every request ought to be answered in ten seconds or less.
3) The webpage ought should open. 3 seconds later, there are more than 100,000 concurrent users.

5. Application and Outcomes

A. Framework

1) Save the collection of data

The user-provided data is saved by the system.

2) Training of models:

The user provides data, which the system enters into the chosen model.n

3) Model forecasts

The user provides the input, and the system uses that data to forecast the outcome.

B. Individual

1) A dataset is loaded

The dataset that the user wishes to process can be loaded.

2) Present the dataset

The dataset is visible to the user.

3) Choose a model

To ensure correctness, the user can apply the template to the dataset.

4) Assessment

The user is able to assess model inefficiencies.

6. Findings and Discussion

We employed supervised\software\training models in this application. To estimate software quality, we employed five machine learning algorithms: logistic regression, bagging classifier, XGBoost classifier, random forest classifier, and tree classifier. All five methods function efficiently and accurately.

A. Approach and Algorithm

1) Classifier XGBoost:

Recently, XGBoost has been the dominant method in both Kaggle and practical machine learning contests for structured or tabular data. XGBoost is a fast and efficient implementation of gradient-enhanced decision trees.

A gradient-enhanced framework is used by the ensemble machine learning algorithm XGBoost, which is based on

decision trees. In unstructured data issues (texts, photos, etc.), ANNs often perform better than any other frameworks or algorithms. Nonetheless, decision tree algorithms are thought to work best with small to medium-sized structured/tabular data. - class at the moment.

Thinking: Assume that there are now several interviewers, each with a distinct voice, rather than just one. Combining information from each interviewer to get a final conclusion through a democratic voting procedure is known as bootstrap or bootstrap pooling.

Both Gradient Boosting Machines (GBM) and XGBoost are compound tree techniques that use gradient descent architecture to boost weak learners (often CARTs). On the other hand, XGBoost makes algorithmic and system enhancements to enhance the GBM core framework.

2) The Random Forest

A machine learning method called Random Forest is used to address classification and regression issues. It makes use of ensemble learning, a method that combines several classifiers to offer answers to challenging issues. Multiple-decision trees make up a random forest algorithm. Using packaging or bootstrap aggregation, the random forest method trains its "forest". An artificial meta-algorithm called sacking increases the precision of machine learning algorithms.

The Random Forest method uses decision tree predictions to decide the outcome. It makes predictions by averaging the results from several trees. The precision of the outcome rises with the number of trees.

The tree pruning algorithm's limitations are eliminated by random forest. Accuracy is increased while data set/inconsistency is decreased. It generates predictions with minimal package requirements (similar to Scikit-learn).

Features of the random forest algorithm:

- Compared to the decision tree algorithm, it is more accurate.
- It offers a productive method for handling missing data.
- A decent predictive hyperparameter setup may be produced by it.
- Decision tree matching is resolved in this way.
- A random selection of items is made from the node distribution point for each random forest tree.

Decision trees are the fundamental building blocks of the random forest algorithm. A tree- like piece of technology used for piggybacking is called a decision tree. An overview of decision trees I would want to know how random forest algorithms work.

A decision tree is composed of three nodes: decision nodes, leaf nodes, and root nodes. A decision tree method is used to split the training data level into branches, which are then subdivided into still more branches. Until a leaf node is encountered, this procedure continues. A leaf node cannot be disconnected at this time.

The nodes of a decision tree represent traits that are utilized to predict an outcome. Decision nodes give paper connectivity. There are three types of decision tree nodes shown. in the diagram that follows

Classifier using Decision Trees:

The tree has several real-world equivalents and has been shown to have an impact on a variety of machine learning domains, such as regression and classification. A decision tree is a visually appealing and understandable way to represent decisions and decision-making in decision analysis. It makes use of a tree-like model, as the name implies. Despite being a frequently used technique in data mining to determine a plan of action to accomplish a particular objective.

The root is at the top of the decision tree, which is depicted upside down. The condition or internal node that determines how the tree is divided into branches and edges is shown by strong black text in the figure on the left. The decision/sheet, where the passenger died or survived, is at the end of the branch, which is no longer separated, and is indicated in red and green lettering, respectively.

You cannot disregard the simplicity of the techniques, even when the real dataset is merely a branch of a larger tree with many additional properties. The links are obvious, and the function's significance is evident. Since the objective is to categorize the passenger as either a survivor or a dead person, our tree is referred to as a classification tree. This technique is more often known as a data learning decision tree. Similar presentation techniques apply to regression trees, with the exception that they forecast continuous quantities like home prices. Classification and Regression Trees, or CARTs, are the usual name for decision tree methods.

The Logistic Regression Model:

Early in the 20th century, the biological sciences employed logistic regression. After then, this was applied widely in social science fields. In cases when the goal, or independent variable, is categorical, logistic regression is employed.

As an illustration,

To determine whether there is spam (1) or (0)

The presence or absence of malignancy in a tumor (1)

Imagine we have to decide whether or not to mark an email as spam. There is no predetermined threshold on which to base categorization if we use linear regression to this problem. Assume the data is classed as non-malignant if the actual class is malignant, the projected constant value is 0.4, and the threshold value is 0.5. This might have dire effects for the data in real time.

It is clear from this example that the classification issue does not lend itself to linear regression. Because it is unrestricted, logistic regression is included in linear regression. They have a strictly 0–1 value range.

Logistic Regression Types:

1) Logistic regression in binary

The conceivable outcomes for a categorical response are limited to two. For instance, spam or not.

2) Logistic multinomial regression

Three or more groups that aren't arranged.

Predicting which food (plant, non-plant, or vegan) is favored more is one example.

3) Regression using Normal Logistic Model three or more groups in order

For instance, Movie Assessment 1–5.

7. Research and Testing for Systems

A. Study of Feasibility

This stage assesses the project's viability and provides a business proposal, a project overview, and some cost projections. The suggested system's viability is investigated as part of the system analysis. This is to make sure the business won't be burdened by the suggested method. The most crucial system requirements must be understood in order to conduct a feasibility study.

Regarding the feasibility analysis, the three primary queries are

- Financial viability
- Technical viability

Financial viability:

The purpose of this study is to confirm the system's financial impact on the company. An organization's financial resources for system research and development have a cap. Expenses need to be warranted. The system was created within the allocated budget as a result, and the majority of the technologies utilized made this possible. All goods that needed to be ordered were customized.

Technical Viability:

The purpose of this research is to confirm that the system meets the technical criteria. The developed systems ought not to place an excessive burden on the technical resources at hand. High demands are placed on the available technological resources as a result. High standards for the client result from this. Because the system's implementation calls for the least amount of resources, the designed system must have modest requirements or remain unchanged.

Social viability:

Verifying user acceptability is a component of the research. It involves the process of instructing the user on how to make efficient use of the technology. The system must be accepted as necessary, and the user must not feel intimidated by it. The techniques employed to inform and acquaint users with the system are the sole factors that determine the degree of user adoption. Because he is the system's final user, he has to have more confidence in order to offer constructive critique.

Examining the system:

Finding bugs is the aim of testing. The process of testing involves looking for any potential weak point or defect in a piece of work. This offers a chance to confirm that parts, subassemblies, assemblies, and/or the final product all perform as intended. It is the practice of utilizing software to make sure the system satisfies both user expectations and its own requirements. and it doesn't fail due to incorrect input. There are different tests. Every test type satisfies a set of requirements.

8. Conclusion

Our research revealed that the accuracy of software quality evaluation is much increased when sophisticated machine learning algorithms are used in conjunction with correlation matrices and feature selection approaches. We have effectively forecasted software quality levels by using a wideset and experimenting with many contemporary approaches including XGBoost, Random Forest, Decision Tree, Logistic Regression, MLP Classifier, and Naive Bayes. This study explains the intricate link between development qualities and software quality and emphasizes the use of machine learning to effectively evaluate software quality.

Upcoming Developments:

Additional estimations for software quality enhancement might involve employing ensemble approaches, which integrate many models to get predictions that are more resilient. Adding deep learning architectures—like recurrent or convolutional neural networks—can also help reveal more details about intricate software quality models. Furthermore, investigating hybrid models that combine cutting-edge machine learning techniques with conventional statistical methods may help to increase accuracy. The evaluation method may be further enhanced by integrating domain-specific functions with research interpretability model decision understanding approaches.

References

[1] Kalaivani, N. and Beena, R. (2018) Overview of Software Defect Prediction Using Machine Learning Algorithms. International Journal of Pure and Applied Mathematics, 118, 3863-3873.

[2] Li, Peng, et al., "An Empirical Analysis of Simplified metric Set-Based Program Defect Prediction." in Information and Software Technology, 59 (2015):170–190.

[3] Yu, Xiao, and colleagues. "Exploiting Class Imbalance Learning from Interfirm Mistakes Predicting." SEKE 2017 is the 29th International Conference on Software Engineering and Knowledge Engineering. Knowledge Systems Institute, KSI Research Inc. ja, 2017.

[4] D. Bowes, T. Hall, and J. Petric, "Program-error prediction: Do different classifiers find the same errors?" in Journal of Software Quality, 26(2), 2018, pp. 525-552.

[5] X. Wang, Y. Zhang, L. Zhang and Y. Shi, "A Knowledge Discovery Case Study of Software Quality Prediction: ISBSG Database," 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Toronto, ON, Canada, 2010, pp. 219-222.

[6] Thomas Zimmermann et al., "A Large-Scale Experiment Data vs. Domain vs. Process for Cross-Project Failure Prediction." The 7th Joint Conference of the ACMSIGSOFT Symposium on Fundamentals of Software Design and the European Conference on Software Engineering is available as proceedings. 2009 ACM.

[7] Amasaki, S., Takagi, Y., Mizuno, O., and Kikuno, Bayesian creating a belief network neural network to forecast the ultimate quality of embedded systems, T. Inf. Syst. IEICE Trans. 8(6), 1134-1141, 2005.

[8] Idri, A., and Abra, A., Measures of fuzzy logic based on project similarity: validation and potential enhancements: Proceedings of the 7th International Symposium on Software Metrics, pp. 85–96. IEEE, UK, England, 2001.

[9] Gopi Krishnan, Rajbahadur, "Effect of Regression Models on the Generation of Error Classifiers" 14 International Conference on Mining Software Archives Proceedings, 2017, IEEE Press.