International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 03 | March - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Performance of Software Quality Prediction with Machine learning Methods

PAINATI ROJA, MTech, Dept. of CSE, Sree Rama Engineering College, Karakambadi Road, Tirupati-517507

Dr.T.HARITHA, M.Tech, Ph.D, Associate Professor, Dept. of CSE, Sree Rama Engineering College, karakambadi Road, Tirupati-517507

ABSTRACT:

The prediction of software quality through machine learning (ML) is an expanding area that focuses on applying different ML algorithms to anticipate the quality of software systems. Software quality assessment is a crucial task required at different phases of software development. It can be utilized for organizing the quality assurance practices of the project and for comparison purposes. In prior studies, two approaches (Multiple Criteria Linear Programming and Multiple Criteria Quadratic Programming) were employed to assess software quality. Additionally, C5.0, SVM, and neural networks were tested for quality assessment. These studies exhibit fairly low accuracy levels. In this research, we sought to enhance estimation precision by utilizing significant attributes from a substantial dataset. We employed a feature selection approach and a correlation matrix to achieve improved accuracies. Moreover, we have tested new techniques that have proven effective for different prediction challenges. Machine learning techniques like XGBoost (Gradient Boosting), Random Forest, Decision Tree, Logistic Regression, Bagging Classifier, and K Nearest Neighbour are utilized on the data to forecast software quality and uncover the relationship between quality and development characteristics. The experimental findings indicate that machine learning algorithms can accurately predict the quality level of software.

Keywords: Prediction of Software Quality, Machine Learning techniques, Development and correlation matrix

Introduction:

Software applications can have faults that arise from requirements analysis, specification, and other tasks performed during the software development process. Consequently, assessing software quality is a necessary task at different phases. It can be utilized for planning project-based quality assurance methods and for setting benchmarks. Moreover, the quantity of defects per unit is regarded as one of the key factors that reflect the quality of the software. Effective quality estimation ultimately contributes to the delivery of higher-quality software that better satisfies user needs and expectations.

In these studies, quality estimation was done by binary classification. We tried to improve these prediction models, taking into account the size in terms of function points and using 4- level classification. We have experimented with recent classification methods shown to be successful for other prediction tasks

Objective of the Study

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.

LITERATURE SURVEY:

Cowlessur, Sanjeev & Pattnaik, Saumendra & Pattanayak, Binod. (2020). A Review of Machine Learning Techniques for Software Quality Prediction. 10.1007/978-981-15-1483- 8_45.

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.

Pattnaik, Saumendra & Pattanayak, Binod. (2016). A survey on machine learning techniques used for software quality prediction. International Journal of Reasonino-based Intelligent Systems. 8. 3. 10.1504/1JRIS.201 6.080058.

In the present software development scenario, software quality prediction has become significantly important for successful implementation of the software in real world application and enhances the longevity of its functionality. Moreover, early identification of anticipated fault prone software modules in the process of development of software is crucial in saving efforts involved in this pro cess. Machine learning techniques are considered to be the most appropriate techniques for software quality prediction and a large spectrum of research work has been conducted in this direction by several authors. In this paper, we conduct an extensive Survey various machine learning techniques like fuzzy logic. Neural network, and Bayesian model, etc. used tor software quality prediction along with an analytical justification for each of the proposed solutions.

Huang, Bing & Li, Xiaojun & Li, Ming & Bernstein, Joseph & Smidts, Carol. (2005). Study of the impact of hardware fault on software reliability. 2005. T0 Pp-. 10.1109/ISSRE.2005.39.

As software plays increasingly important roles in modern society, reliable software becomes desirable tor all stakeholders. One of the root causes of software failure is the failure of the computer hardware platform on which the software resides. Traditionally, fault injection has been utilized to study the impact of these hardware failures. One issue raised with respect to the use of fault injection is the lack of prior knowledge on the faults injected, and the fact that, as a consequence, the failures observed may not represent actual operational failures.

SYSTEM ANALYSIS

EXISTING SYSTEM:

In the previous studies, two methods (Multiple Criteria Linear Programming and Multiple Criteria Quadratic Programming) for estimating the quality of software had been used.

Also, C5.0, SVM and Neutral network were experimented with for quality estimation. These studies have relatively low accuracies.

The quality level was classified according to: number of minor defect + 2*number of major defect + 4*number of extreme defect. The quality of level was to be either high or low. They used k-fold cross-validation technique to measure MCLP and MCỘP's performance on the ISBSG database. Release 10 Dataset which contained 4.017 records and 106 attributes was used. After pre-processing, 374 records and 11 attributes remained in the dataset.

Limitations of existing systems

- Lack of accuracy
- Difficult to Handle.

PROPOSED SYSTEM

In this study, we aimed to improve estimation accuracy by using relevant features of a large dataset. And KNN model proposed in this project for estimating the software quality. We used a feature selection method and correlation matrix for reaching higher accuracies. In addition, we have experimented with recent methods shown to be successful for other prediction tasks. Machine learning algorithms such as XgBoost, Random Forest, Decision tree. Logistic Regression and Bagging Classifier are applied to the data to predict the software quality.

ADVANTAGES OF PROPOSED SYSTEM:

It is very much faster than manual system. It is easy and fastest record finding technique. It is very much flexible to work. Man power required is very less.

1.High accuracy.
2.Time Saving.
3.Low complexities.
4.High reliability.

MACHINE LEARNING ALGORITHMS:

XGBoost Classifier: XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.

XGBoost and Gradient Boosting Machines (GBMS) are both ensemble tree methods that apply the principle of boosting weak learners (CARTs generally) using the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimization and algorithmic enhancements.

Random Forest Classifier:

Random Forest algorithm is a supervised classification algorithm. We can see it from its name, which is to create a forest by some way and make it random. There is a direct relationship between the number of trees in the forest and the results it can get: the larger the number of trees, the more accurate the result. But one thing to note is that creating the forest is not the same as constructing the decision with information gain or gain index approach.

Decision Tree Classifier:

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal.

Logistic Regression: Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable (target) is categorical.

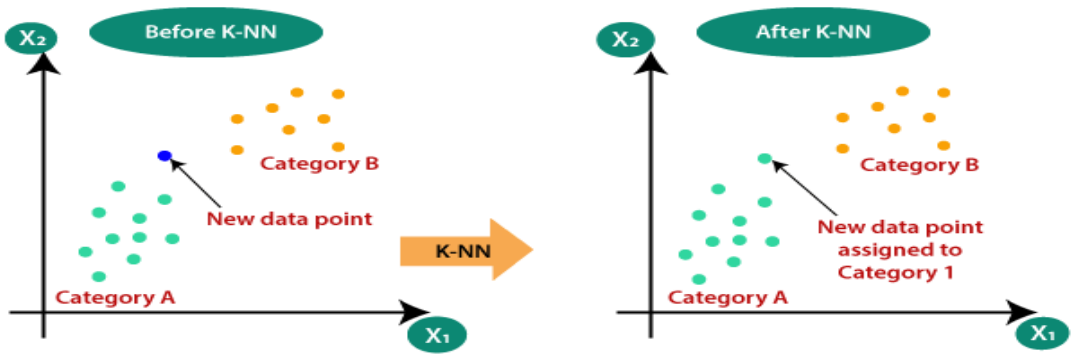
Bagging Classifier:

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g- a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

K-Nearest Neighbour:

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point Xl, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class or a particular dataset. Consider the below diagram.



K-NN Algorithm Data set

IMPLEMENTATION OF SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution.

Software system tests if the system meets the specified requirements and if it is suitable for delivery to the end-users and does not fail in an in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing: Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black Box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated. as a black box -you cannot "'see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing: Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach: Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing: Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. software applications at the company level - components in a software system or - One step up interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing: User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

RESULTS AND CONCLUSION:

We have experimented classification algorithms sing Scikit-learn library on two dataset. We have experimented with recent algorithms that support multi-class classification.

Results of ISBSG Dataset

ALGORITHM	ACCURACY
Bernoulli NB	69.52%
Decision Tree Classifier	89 43%
Random Forest Classifier	8943%
XGB Classifier	92. 28%
Bagging Classifier	92.28%

Results of EBSPM Dataset

ALGORITHM	ACCURACY
XGB Classifier	65.56%
Random Forest Classifier	66.67%
Bagging Classifier	67.78%
Decision Tree Classifier	67.78%
Logistic Regression	92.22%

The accuracies achieved by using these algorithms are 92.28% on EBSPM Data set and 92.229% On ISBSG Dataset. In comparison to previous directly comparable studies, acceptable level multiclass quality prediction could be achieved.

We have used supervised Machine Learning models to predict the quality of the software, We used five ML algorithms to predict software quality they are Random Forest Classifier, Decision Tree Classifier, XGBoost Classiier, Logistic Regression and Bagging Classifier. All five algorithms perform well with good accuracies.

REFERENCES

- N.Kalaivani, Dr.R.Beena, International Journal of Pure and Applied Mathematics Volume 118 No, 20 2018, 3863-3873 ISSN: 1314-3395.
- He, Peng, et al. "An empirical study on software defect prediction with a simplified metric set. " Information and Software Technology 59 (2015): 170-190
- Yu, Xiao, et al. "Using Class Imbalance Learning for Cross-Company Defect Prediction." 29th International Conference on Software Engineering and Knowledge Engineering (SEKE 2017). KSI Research Inc. and Knowledge Systems Institute, 2017.
- D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: do different classifiers find the same defects?." Software Quality Journal, 26(2), 2018, pp. 525-552
- X. Wang, Y. Zhang, L. Zhang and Y. Shi, "A Knowledge Discovery Case Study of Software Quality Prediction: ISBSG Database," 2010 IEEE/VWIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Toronto, ON, 2010, pp. 219-222.
- X. Wang, Y. Zhang, L. Zhang and Y. Shi, "A Knowledge Discovery Case Study of Software Quality Prediction Based on Classification Models: ISBSG Database," The 11th International Symposium on Knowledge Systems Sciences (KSS 2010), 2010
- Zimmermann, Thomas, et al. "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process." Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering. ACM, 2009.
- 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
- ldri, 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)
- Rajbahadur, Gopi Krishnan, et al. "The impact of using regression models to build defect classifiers." Proceedings of the 14th International Conference on Mining Software Repositories. IEEE Press, 2017.