JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.2175-2182
https://publishoa.com
ISSN: 1309-3452

Prediction of Software Quality Using Machine Learning

1. **Shaik Mohammed Ilias**, Assistant Professor of CSE, St. Peter's Engineering College, Telangana, India,ilias@stpetershyd.com
2. **Javvaji Niharika**, Student (4thB. Tech, CSE), St. Peter's Engineering College, Telangana, India, javvajiniharika5@gmail.com
3. **GaddamHarshavardhan Reddy**, Student (4thB. Tech, CSE), St. Peter's Engineering College, Telangana, India, harshareddy7230@gmail.com
4. **GopuSrujan**, Student (4thB. Tech, CSE), St. Peter's Engineering College, Telangana, India,nanisrujan3147@gmail.com

Received 2022 April 02**; Revised** 2022 May 20**; Accepted** 2022 June 18

Abstract: Software programmes may have flaws resulting from requirements study, definition, and other software development operations. Applications' management and quality regulation would benefit immensely from programmers being able to predict the quality of early-stage apps. MCQP and MCLP(Multiple Criteria Quadratic Programming and Multiple Criteria Linear Programming) were the two techniques employed in early studies to determine the programme quality. By utilising pertinent information from a sizable dataset, we attempted to increase estimation accuracy in this paper.Random Decision Forest, Gradient Boosting, Decision Tree Classifier, Naive Bayes Classifier, and other machine learning algorithms are used to analyse the data and forecast software quality andto demonstrate the connection between the improvement and quality attributes. The outcomes of the experiment demonstrate the level of software quality.

Keywords: – Software Quality, Machine Learning,Gradient Boosting, Training, Predictions

I. Introduction:

Software has become increasingly important as information technology has advanced, particularly in highly distributed communication networks, such as those used in the army, aviation, and banking institutions [4]. Therefore, software quality prediction is a challenge required at various stages. [1]. To minimise the quantity of effort necessary for this process, it is essential to identify anticipated fault-prone software modules early on [13]. It can be used for benchmarking as well as planning project-based quality assurance practises. Additionally, one of the most crucial indicators of the software's quality is

thought to be the quantity of flaws per unit [2]. The domain of error estimation has already seen the growth in various methods and models [5]. Additionally, if the software system is huge, countless software flaws will be discovered during the testing phase. However, the number of flaws discovered and resolved through debugging actions keeps changing very little, in contrast to the failure content at the beginning of the testing phase [10]. Because of their potentially severe effects

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.2175-2182
https://publishoa.com
ISSN: 1309-3452

on computer security and information safety, exploitable software flaws have received a lot of attention lately [9]. The evaluation of the ERM is a crucial task in the creation of software [7]. To assist various user types with quality issues, a number of models have been proposed [14]. They combined a neural network, SVM and C5.0, SVM, and a neutral network for classification. In a more practical study, Rashid et al. used case-based reasoning (CBR) for software quality estimation. CBR is a model for machine learning that learns by using the outcomes of previous experiments. By using 4-level classification and accounting for the size of function points, we attempted to enhance these prediction models. We tested recent classification techniques that have been effective for other prediction tasks.

The organization of this article is in following manner i.e., Section-II describes the research background, where we reviewed and analysed about all the literatures, Section-III denotes about the existing regime, Proposed methodology explained in Section-IV, results and discussions are demonstrated in Section-V and finally conclusions are noted in Section-VI.

II. Research Background:

Observation1:

A formal process for assessing and recording the efficiency of the work products at every stage of the development process life cycle is known as software quality assurance, according to the research paper titled "Software quality metrics in quality assurance to study the impact of external factors related to time" by the authors D. H. Done, T. John, D. M. G. Chand, and Vijay[1]. Customer value evaluation problems are resolved in this paper by: Create a framework using various software quality standards. describes software metrics. There are one or even more Quality assurance quality measure metrics for each action with in software application lifecycle that are concerned with ensuring the high calibre of the procedure and the finished product. measures of quality for future software projects.To achieve this, the appropriate methodology for applying software quality metrics with software quality assurance is used in the quality life cycle is used.

Observation2:

According to a study by T. Hall, J. Petric and D. Bowes titled "Software defect prediction: Do different classifiers find the same defects?", numerous models for predicting defects have indeed been published. over the past ten years [2]. According to the reports, Comparable classifier performance is used in these models, with models infrequently outperformingthe maximum recall rate for predictions of around 80%. Using sensitivity analysis, we compare the ability of the SVM classifiers, RPart, Naive Bayes and Random Forest to identify flaws in Space agency, open-source, and commercial datasets. A confusion matrix is used to represent the defect predictions made by each classifier,and comparisons are made between each classifier's prediction uncertainty.

Observation3:

Software is becoming increasingly significant in contemporary society, according to a study by authors Y. Shi, Y. Zhang, X. Wang, L. Zang titled "A Knowledge Discovery Case Study of

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.2175-2182
https://publishoa.com
ISSN: 1309-3452

Software Quality Prediction: ISBSG Database" [3]. In order to forecast software quality andexplain the connection between software quality and development attributes, this study applies the MCLP model to the ISBSG database. The experimental finding demonstrates that the Multiple Criteria Linear Programming(MCLP) Model can accurately forecast the software quality level. A number of helpful inferences have also been made from the results of the experiment.

Observation4:

Over time, in close cooperation with software developers from the Netherlands and Belgium, we developed the Evidence-Based Software Portfolio Management (EBSPM) tool, is described and evaluated in the research paper titled "Evidence-based software portfolio management" by the author H. Huijgens. The EBSPM-tool aims to encourage innovation in a business's capacity for software deliveryby measuring, analysing, and benchmarking the effectiveness of linked sets of software systems in terms of scope, budget, schedule, and defect count [8]. We demonstrate that the EBSPM-tool, particularly with regard to its benchmarking and visualisation goals, can be successfully used in an industrial context.

III. Existing Regime:

Software quality is a difficult idea to grasp. As a result, both in practise and research [11], evaluating and forecasting it remains difficult. An essential step in ensuring the sufficiency of software quality products is the evaluation of software quality, which can be done with the aid of a software quality model [12].Two studies on predicting software quality using defect quantities in the ISBGS (The InternationalSoftware Benchmarking Standards Group) dataset are comparisons are made [6]. In the first study, the dataset was used to test the two methods (MCQP and MCLP), and the outcomes were compared [3]. Either a high- or low-quality level was expected. If the software meets the following criteria—those extreme defects exist, that there is more than one major defect, or more than ten minor defects, it is classified as high quality. The remaining ones are considered to be of lower quality. On the ISBSG database, they evaluated the performance of MCLP and MCQP using the k-fold cross-validation technique. They used (the January 2007-released) Release 10 dataset, which contained 4,017 records and 106 attributes. After pre-processing, the dataset still contained 374 records and 11 attributes.

IV. Proposed Methodology:

In order to predict software quality, our proposed system makes use of a multiple algorithms in machine learning, including CNN, Bagging Classifier, Random Forest Classifier, Gradient Boosting, Decision Tree Classifier, Bernoulli NB, and Logistic Regression. The author used two datasets to carry out this project, but since those datasets aren't online, I'm only using the one the student sent. The Dataset folder contains this dataset.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.2175-2182
https://publishoa.com
ISSN: 1309-3452

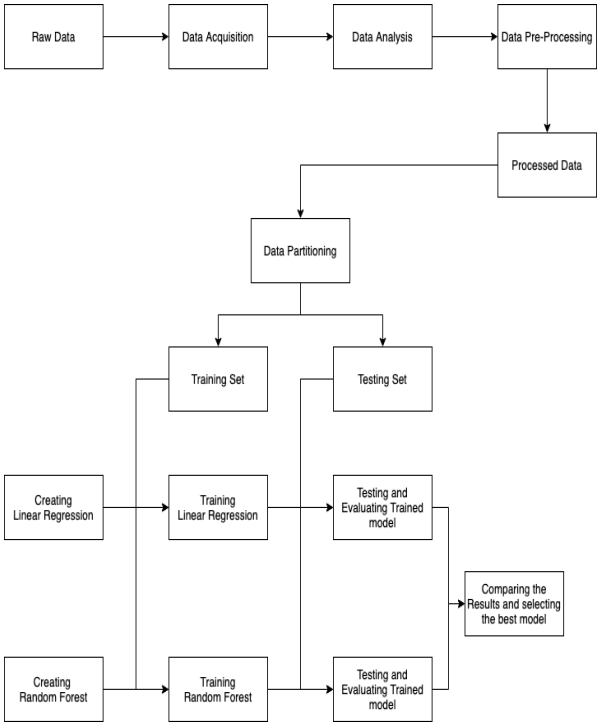

Fig 4.1: Flow Chart for the Suggested Methodology

The process illustrated in Figure 4.1 above explains itself as follows:

1.Data Acquisition: In data acquisition,data that can be used for training and testing is downloaded from an online source.

2. Data Analysis: Using graphical representations and brief statistics and while understanding the fundamentals of the loaded data, we perform a preliminary analysis of the information in order to find patterns, identify anomalies, evaluate hypotheses, and validate assumptions.

3. Data Pre-Processing: Data pre-processing transforms raw information into a form that computer systems and machine learning algorithms can comprehend and evaluate as part of the data mining and analysis process.

4. Data Splitting: The validation, training and testing data sets are the three non-overlapping sets that make up the data splitting process in data mining.

5. Test Set: The subset of data that was used to impartially assess how well a final model fit the training dataset.

6. Train Set: It is a subset of the data set that is used to fit a model for the classification or prediction of values that are known in the training set but unknown in additional (future) data.

a) Instantiating the algorithms: To begin the training, multiple algorithms that can accept input and produce output are instantiated and given the train data. These are the algorithms this methodology employs:

i. Naïve Bayes Algorithm:

The supervised learning algorithm known as the Naive Bayes algorithm, which is founded on the Bayes principle, is primarily used to classify texts.

P(V|W) equals P(V)*P(W|V)/P (W). ——-- Equation-1

Where,

• Prior likelihood (P(W)) of W
• The earlier likelihood of class V is P(V).
• P(W|V) is the likelihood that predictor B will occur given class A probability.

ii. Decision Tree Algorithm:

The supervised learning algorithm known as the Naive Bayes algorithm, which is founded on the Bayes principle, is primarily used to classify texts.

Random Forest Algorithm:

A component of the supervised learning approach is Random Forest. It can be used for ML issues involving both regression and classification.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.2175-2182
https://publishoa.com
ISSN: 1309-3452

Gain (D, A) = Entropy (D), Entropy (D, A) ———- Equation-2

Entropy (D, A) is the entropy determined following the data's division based on feature A

where

- D is the desired variable
- The feature to be divided on is A, and
- The entropy determined after the data is divided based on feature A is called entropy (D, A).

iv. Logistic Regression Algorithm:

Essentially, supervised classification is what logistic regression does.In a classification task, the desired variable (or outcome), D, could only take distinct values for a particular feature set (or inputs), A. Despite what most people think, a regression model includes logistic regression.

v. Bagging Classifier Algorithm:

A bagging classifier is a combination of meta-estimator that aligns classification model one at a time to arbitrary subgroups of the original dataset, then combines the individual forecasting (either by casting a vote or by averaging) to create a final prediction.

vi. Gradient Boosting Algorithm:

One well-liked enhancing algorithm is gradient boosting. In gradient boosting, each predictor rectifies the inaccuracy of its forerunner.

vii. Convolutional Neural Network:

The processing of structured arrays of data using a deep learning neural network, such as portrayals, is known as a convolutional neural network, or CNN.

7. Training: Making a specific algorithm comprehend the training data and develop concept intelligence.

8. Testing: The Processis employed to assess the effectiveness of the trained model and predict the outputs for the inputs in the test set.

9. Comparing: The Process is used to evaluate the effectiveness of every algorithm and draw a conclusion. In this case, we use accuracy, which is determined by the formula:

Accuracy equals 100% - Rate of Error------ Equation-3

Rate of Error is measured by mod of the difference betweendivided by Actual Value multiplied by 100-----Equation-4

V. Results Analysis:

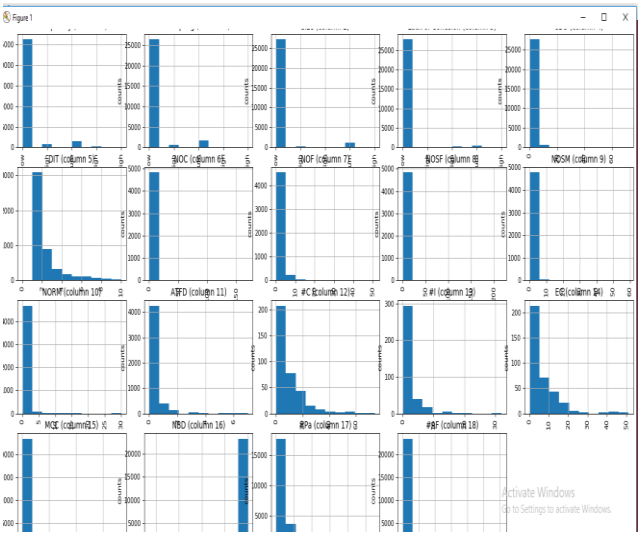

Fig 5.1:A graph displaying the dataset's columns

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.2175-2182
https://publishoa.com
ISSN: 1309-3452

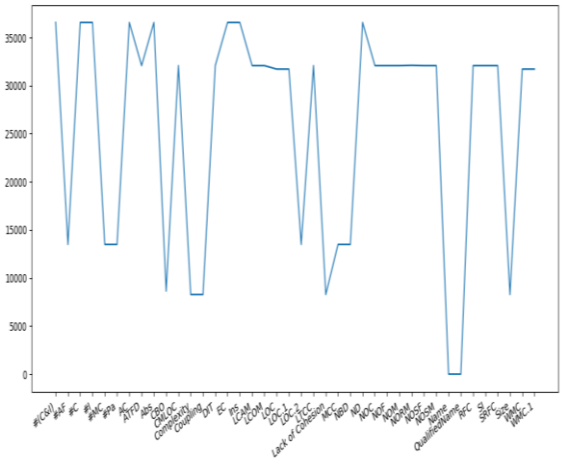

Fig 5.2: A graph showing the total number of missing values for each column.

Fig 5.3: Following application of the feature selection algorithm, features and records

Fig 5.4: Precision, accuracy, recall, and fscore for Naïve Bayes,Logistic Regression, Random ForestClassifier, Decision Tree Classifier

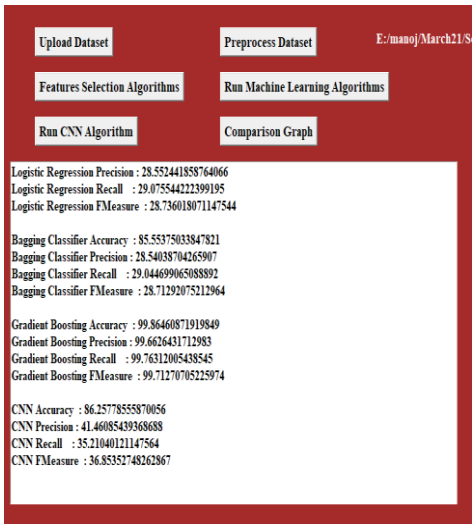

Fig 5.5: Accuracy, precision, recall, and fscore for CNN, Bagging Classifier, and Logistic Regression

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p.2175-2182
https://publishoa.com
ISSN: 1309-3452

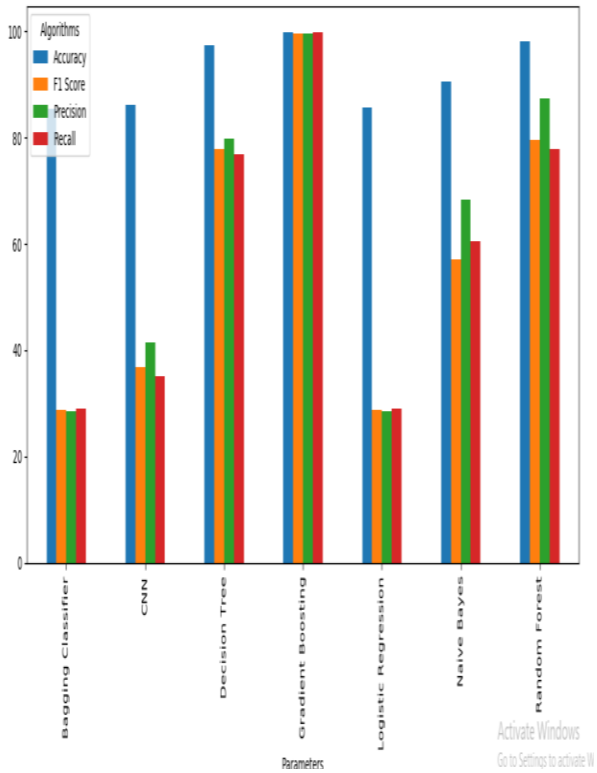

Fig 5.6: A graph plotting accuracy, precision, and recall

VI. Conclusion:

Early modern algorithms for multi-class classification have been tested by us. These algorithms achieved 92.28 percent accuracy on the Evidence-Based Software Portfolio Management (EBSPM) dataset and 92.22 percent accuracy on the International Software Benchmarking Standards Group (ISBSG) dataset. Multiclass quality prediction at an acceptable level could be accomplished when compared to earlier, identical studies.

The quality of the software developed is solely responsible for theeffective use of a software product. However, a software developer faces significant difficulties when attempting to estimating the software product's quality before it is put into use in practical applications. According to the literature, there has only been a small numberof studies in this field reported so far.

The majority of researchers have focused their work on utilising a variety of machine learning methods to predict software quality. Another factor relating the ability to predict software quality is that in order to minimise the developer's workload while creating a software product, the forecasting must be formed early on in the software development process. In this paper, we conducta thorough examination of machine learning approaches used to forecast quality of the software.

Acknowledgement:

We appreciate St. Peter's Engineering College's laboratory assistance and ongoing support in helping us prepare this paper in a more enlightened manner.

References:

[1] Vijay, T. John, D. M. G. Chand, and D. H. Done. "Software quality metrics in quality assurance to study the impact of external factors related to time." International Journal of Advanced Research in Computer Science and Software Engineering, 2017.

[2] D. Bowes, T. Hall, and J. Petrić, "Softwaredefect prediction: do different classifiers find the same defects?." Software Quality Journal, 26(2), 2018, pp. 525-552.

[3] 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, 2010, pp. 219-222.

[4] 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

[5] E. Rashid, S. Patnaik, and V. Bhattacherjee, "Software quality estimation using machine learning: Case-Based reasoning technique, " International Journal of Computer Applications, 2012

[6]"International Software Benchmarking Standards Group (ISBSG)",[online] Available:www.isbsg.org

[7] Moody, D.L., 2003,"Measuring the quality of data models: an empirical evaluation of the use of quality metrics in practice," ECIS 2003 Proceedings 78.

[8] H.Huijgens,"Evidence-based software portfolio management: a tool description and evaluation", 20th International Conference on Evaluation and Assessment in Software Engineering (EASE '16), 20

[9] Ghaffarian, S. M., &Shahriari, H. R. (2017),"Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey. ACM Computing Surveys", (CSUR), 50(4), 1-36.

[10]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)

[11] Wagner, S: "A bayesian network approach to assess and predict software quality using activitybased quality model." Inf. Softw. Technol. 52(11), 1230–1241 (2010)

[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] 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)

[14]P. Miguel, J., Mauricio, D., Rodríguez, G., 2014. "A Review of Software Quality Models for the Evaluation of Software Products. International Journal of Software Engineering & Applications", 5, 31–53. https://doi.org/10.5121/ijsea.2014.5603