JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Predicting Software Quality Using Machine Learning

Mr. Yerranagu Sai Kiran (M.C.A). Rajeev Gandhi Memorial college Of Engineering and Technology, Nandyal

***Mr. P. Naveen Sundar Kumar MTech. Rajeev Gandhi Memorial college Of Engineering and Technology, Nandyal**

Abstract

Various stages of the software development process call for the activity of software quality estimate. It could be utilized for benchmarking and planning the project's quality assurance procedures. Multiple Criteria Linear Programming and Multiple Criteria Quadratic Programming were the two techniques employed in early studies to determine the program quality. Additionally, experiments using C5.0, SVM, and Neutral networks for quality estimation were conducted. These studies have a paltry degree of accuracy. In this study, we used pertinent information from a huge dataset to increase estimation accuracy. To achieve improved accuracy, we used a feature selection method and a correlation matrix. We have also tested more contemporary techniques that have been effective for various prediction challenges. Algorithms for machine learning like XGBoost, Random Forest, and Decision Tree To forecast software quality and identify the relationship between quality and development parameters, data is processed using logistic regression and naive bayes. These algorithms produce low recall and precision ratings. We employ the cat boost algorithm and the SMOTE approach to get around this. This suggested technique outperforms the current approach while also scoring well in terms of recall and precision.

Keywords: Estimation, Machine Learning, Software Quality, Extreme Gradient Decent, Boosting

1. INTRODUCTION

1.1 Introduction

Software programs may have flaws resulting from requirements study, definition, and other software development operations. Estimating software quality is hence a task required at various points. It can be used for benchmarking as well as planning project-based quality assurance methods. In addition, one of the most crucial indicators of software quality is thought to be the quantity of flaws per unit.The performance of a system on which software is implemented, including execution time, memory usage, error probability, etc., can be used to describe the quality of a software product. Additionally, the effort put forth by the software developer also plays a significant role in determining the level of quality of a software product. A software product's quality can be regarded as both internal and external. A software's internal quality can be evaluated throughout the software development life cycle (SDLC), whereas its external quality can be measured during implementation and evaluated in terms of functionality. Its interior quality affects both its exterior and internal qualities. Quality models that describe a function of the internal quality attributes can be created in order to evaluate the exterior quality of a software product. To do this, it is necessary to first identify the internal quality attributes, followed by the relationship between the internal and external quality attributes. A number of methods for predicting software quality have been put out by different authors. However, as claimed by the machine learning approach to creating such a model, it looks to be more popular and more successful..

2. Literature Survey

• **[1] Cowlessur, Sanjeev & Pattnaik, Saumendra & Pattanayak, Binod. (2020). A Review of Machine Learning Techniques for Software Quality Prediction:**, The quality of the software generated is solely responsible for the successful deployment of a software product. However, the software developer has substantial difficulties when attempting to anticipate the quality of a software product before it is put into use in practical situations. There has only been a little amount of research in this field documented so far, according to the literature. The majority of researchers have focused their attention on applying various machine learning approaches to forecast software quality.

[2] A. A. CERAN and Ö. Ö. TANRIÖVER, "An experimental study for software quality prediction with machine learning methods," 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications: Estimating software quality is a task required at different phases of the software development process. It could be utilized for benchmarking and planning the project's quality assurance procedures. Multiple Criteria Linear Programming and Multiple Criteria Quadratic Programming were two techniques that have been utilized in prior studies to gauge the quality of software. Additionally, experiments using C5.0, SVM, and Neutral networks for quality estimation were conducted..

[3] Pattnaik, Saumendra & Pattanayak, Binod. (2016). A survey on machine learning techniques used for software quality prediction. International Journal of Reasoning-based Intelligent Systems

Software quality prediction has become crucial in the current software development environment for the successful implementation of the software in real-world applications and for extending the functionality of that program over time. Additionally, early detection of software modules that are expected to have faults is essential for saving time and effort during the software development process. The most effective methods for predicting software quality are thought to be machine learning approaches, and several authors have undertaken extensive study in this area. In this study, we do a thorough examination of different machine learning approaches for software quality prediction, including fuzzy logic, neural networks, Bayesian models, etc., and provide an analytical explanation for each suggested solution.

3. OVERVIEW OF THESYSTEM

3.1 Existing System

in the current framework. To achieve improved accuracy, we used a feature selection method and a correlation matrix. We have also tested more contemporary techniques that have been effective for various prediction challenges. In order to estimate the software quality, machine learning methods with low precision and recall are applied to the data, including XGBoost, Random Forest, Decision Tree, Logistic Regression, and Naive Bayes.

3.1.1 Disadvantages of Existing System

- Less feature compatibility
- Low accuracy.

3.2 Proposed System

https://quillbot.comTo address the issue of class imbalance in the proposed system, we combined the Cat boost machine learning model with the SMOTE technique. Our suggested model surpasses the current approaches and has strong precision and recall scores.

3.3 Methodology

In this project work, I used five modules and each module has own functions, such as:

1. System Module
2. User Module

1. System:

Store Dataset:

The System stores the dataset given by the user.

Model Training:

The system takes the data from the user and fed that data to the selected model.

Model Predictions:

The system takes the data given by the user and predict the output based on the given data.

2. User:

Load Dataset:

The user can load the dataset he/she want to work on.

View Dataset:

The User can view the dataset.

Select model:

User can apply the model to the dataset for accuracy.

Evaluation:

User can evaluate the model performance.

ALGORITHMS:

XGBoost Classifier: XGBoost is an algorithm that has recently dominated Kaggle competitions for structured or tabular data as well as applicable machine learning competitions. A gradient boosted decision tree implementation created for speed and performance is called XGBoost. A gradient boosting framework is used by the ensemble machine learning method XGBoost, which is decision-tree based. Artificial neural networks frequently outperform all other algorithms or frameworks in prediction issues involving unstructured data (pictures, text, etc.). However, decision tree-based algorithms are currently thought to be best-in-class for small- to medium-sized structured/tabular data.Bagging: Assume that there is now a panel of interviewers, each of whom has a vote, as opposed to just one. Using a democratic voting procedure, bagging or bootstrap aggregating combines the information from each interviewer to determine the final outcome. Both XGBoost and Gradient Boosting Machines (GBMs), ensemble tree approaches, use the gradient descent architecture to boost weak learners (CARTs in general). But XGBoost enhances the fundamental GBM architecture with system optimization and algorithmic improvements.

4 Architecture

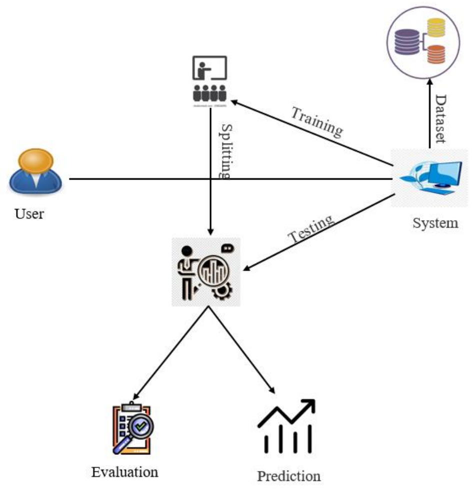
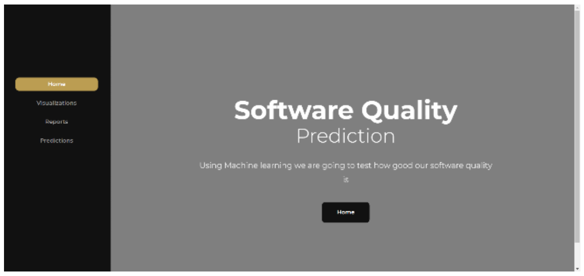


Fig 1: Frame work of proposed method

5 RESULTS SCREEN SHOTS

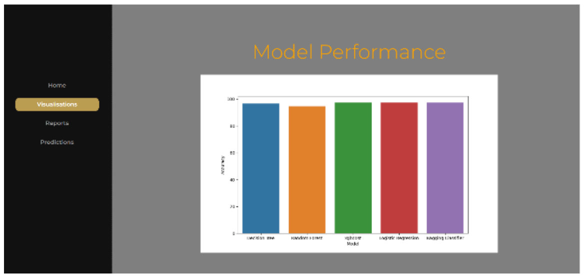
Home Page:



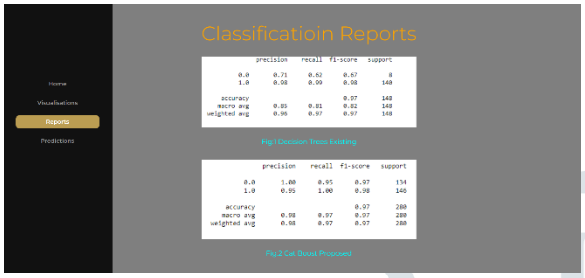
Software Quality Prediction

Using Machine learning we are going to test how good our software quality is

Upload image:



Choose options:



```
              precision    recall  f1-score   support

         0.0       0.71      0.62      0.67         8
         1.0       0.98      0.99      0.98       140

    accuracy                           0.97       148
   macro avg       0.85      0.81      0.82       148
weighted avg       0.96      0.97      0.97       148
```

Fig:1 Decision Trees Existing

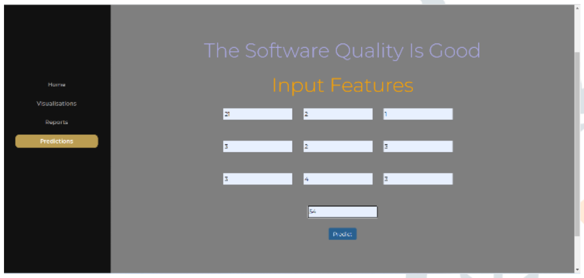
```
              precision    recall  f1-score   support

         0.0       1.00      0.95      0.97       134
         1.0       0.95      1.00      0.98       146

    accuracy                           0.97       280
   macro avg       0.98      0.97      0.97       280
weighted avg       0.98      0.97      0.97       280
```

Fig.2 Cat Boost Proposed

Predict Result:



21	2	1
3	2	3
3	4	3
54		

7. CONCLUSION

✓ To forecast the software quality in this application, we used the supervised machine learning model cat boost with the SMOTE approach. When compared to other machine learning algorithms, the cat boost approach performs well.

8. References

[1] N.Kalaivani, Dr.R.Beena, International Journal of Pure and Applied Mathematics Volume 118 No. 20 2018, 3863-3873 ISSN: 1314-3395.

[2] He, Peng, et al. "An empirical study on software defect prediction with a simplified metric set." Information and Software Technology 59 (2015): 170-190 .

[3] 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.

[4] 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

[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, 2010, pp. 219-222.

[6] 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

[7] Zimmermann, Thomas, et al. "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process." Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. ACM, 2009.

.