AN EXPERIMENTAL STUDY FOR SOFTWARE QUALITY PREDICTION USING MACHINE LEARNING METHODS

Dr. A. Nagaraju [1]., J. Nikhil [2] ., K. Nikhil [3]., N. Karthik [4].,CH. Sai Kapil Varma [5.,]

[1] Head of Department, [2,3,4,5]Students B. Tech -IT, (21S15A1205, 21S15A1206, 21S15A1202,20S11A1248).

Malla Reddy Institute of Technology and Science., Maisammaguda., Medchal., Ts, India

[1]dranraju.mrits@gmail.com , [2]japalanikhil372@gmail.com , [3]kosinanikhil@gmail.com , [4]mudirajkarthik610@gmail.com ,[5]saikapil934@gmail.com , .

ABSTRACT

Software quality estimation is an activity needed at various stages of software development. It may be used for planning the project`s quality assurance practices and for benchmarking. In earlier 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. In this study, we aimed to improve estimation accuracy by using relevant features of a large dataset. 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 Boost, Random Forest and Decision Tree are applied to the data to predict the software quality and reveal the relation between the quality and development attributes. The experimental results show that the quality level of software can be well estimated by machine learning algorithms.

1. INTRODUCTION

Software applications may contain defects, originating from requirements analysis, specification and other activities conducted in the software development. Therefore, software quality estimation is an activity needed at various stages . It may be used for planning project-based quality assurance practices and for benchmarking. In addition, the number of defects per unit is considered one of the most important factors that indicate the quality of the software . There are two directly comparable studies on software quality prediction using defect quantities in ISBGS dataset. In the first study, the two methods (MCLP and MCQP) were experimented with the dataset and the results were compared . The quality level was classified according to: number of minor defects + 2*number of major defects + 4*number of extreme defects. The quality of level was to be either high or low. They used k-fold cross-validation technique to measure MCLP and MCQP's performance on the ISBSG database. Release 10 Dataset (released in January 2007) which contained 4,017 records and 106 attributes was used. After preprocessing, 374 records and 11 attributes remained in the dataset. In another study, the same data set was used again .[The software belonged to high quality class if it fulfils the following requirements: the extreme defects exist or the number of major defects is more than 1 or the number of minor defects is more than 10. The rest are assumed to belong to low quality class. After preprocessing, 746 projects and 53 attributes remained in the dataset. They used C5.0, SVM and Neutral network for classification. As an example, to a more application-oriented study Rashid et al. used case-based reasoning (CBR) for software quality estimation. CBR is a machine learning model which performs the learning process using the results of the previous experiments. Line of code, number of functions, difficulty level, and development type and programmers experience are entered and these attributes are used for estimation. The deviation is calculated by using Euclidian distance (ED) or The Manhattan distance (MD). If the error in estimation is less than 10% then the record is saved to the database. Number of inputs that can be obtained from the user is limited. Also, it is necessary to have close values in the database in order to estimating precise values. 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.

2. LITERATURE SURVEY

1.Software defect prediction (SDP) plays a key role in the timely delivery of good quality software product. In the early development phases, it predicts the error-prone modules which can cause heavy damage or even failure of software in the future. Hence, it allows the targeted testing of these faulty modules and reduces the total

development cost of the software ensuring the high quality of end-product. Support vector machines (SVMs) are extensively being used for SDP. The condition of unequal count of faulty and non-faulty modules in the dataset is an obstruction to accuracy of SVMs. In this work, a novel filtering technique (FILTER) is proposed for effective defect prediction using SVMs. Support vector machine (SVM) based classifiers (linear, polynomial and radial basis function) are designed utilizing the proposed filtering technique over five datasets and their performances are evaluated. The proposed FILTER enhances the performance of SVM based SDP model by 16.73%, 16.80% and 7.65% in terms of accuracy, AUC and F-measure respectively.

2. The purpose of this research is to build a software quality assessment model to evaluate the quality of mobile-based elderly fall detection software. The assessment is based on the quality factors found in the software quality model. The quality factor is adjusted to the characteristics of the software. The model is needed because the software has its own characteristics. This research consists of several stages. The first thing to do is analysing the software domain to determine its characteristics. The second is defining the software assessment needs by mapping software characteristics with the quality standards used (ISO / IEC 25010: 2011) to obtain the appropriate quality factors. The software quality metrics is determined after the quality factors is obtained. The metric to be used is Goal Question Metrics (GCM). The third is software quality weighting process, including its criteria and sub-criteria. Determination of the equation for software quality assessments is the final stage of the research. Based on the research process, it can be concluded that the model developed successfully can be used to assess the software.

3. During the last 10 years, hundreds of different defect prediction models have been published. The performance of the classifiers used in these models is reported to be similar with models rarely performing above the predictive performance ceiling of about 80% recall. We investigate the individual defects that four classifiers predict and analyse the level of prediction uncertainty produced by these classifiers. We perform a sensitivity analysis to compare the performance of Random Forest, Naïve Bayes, Part and SVM classifiers when predicting defects in NASA, open source and commercial datasets. The defect predictions that each classifier makes is captured in a confusion matrix and the prediction uncertainty of each classifier is compared. Despite similar predictive performance values for these four classifiers, each detects different sets of defects. Some classifiers are more consistent in predicting defects than others. Our results confirm that a unique subset of defects can be detected by specific classifiers.

4. Software defect prediction plays an increasingly critical role in emerging software systems. However, existing software defect prediction approaches typically suffer from low accuracy due to the under/over fitting problems. To address this problem, we propose an ensemble learning approach to achieve the accurate defect prediction, where various machine learning algorithms, i.e., artificial neural network, random forest, k-nearest neighbour methods are integrated together. The proposed software defect prediction workflow is introduced. Experiments are conducted to verify the effectiveness of the proposed method. Extensive experiment results verify that our proposed method can improve the defect prediction accuracy when compared with existing methods.

3.METHODOLOGY

Existing System:

Software quality estimation is an activity needed at various stages of software development. It may be used for planning the project`s quality assurance practices and for benchmarking. In earlier 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

Proposed System:

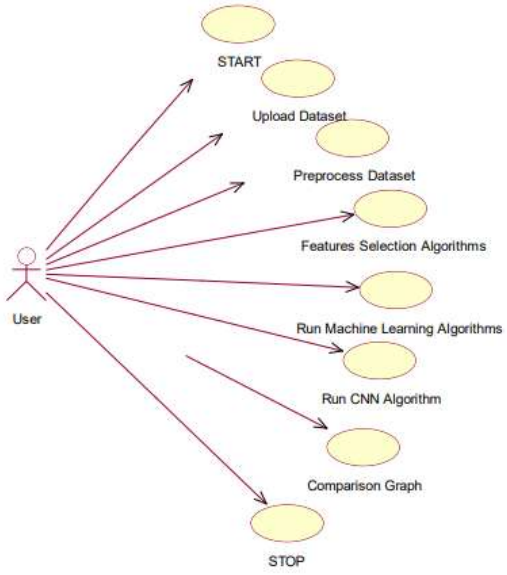
In this paper We used a feature selection method and correlation matrix for reaching higher accuracies. In addition, we have experimented with recent methods shown to be successful for other prediction tasks. Machine learning algorithms such as Xboost , Random Forest and Decision Tree are applied to the data to predict the software quality and reveal the relation between the quality and development attributes. The experimental results show that the quality level of software can be well estimated by machine learning algorithms.

UML DIAGRAMS

UML (Unified Modeling Language) is a standardized language for specifying, visualizing, constructing, and documenting the artifacts of a software system, as well as business modeling and other non-software systems. The goal of UML is to provide a common language for creating models that are understandable to developers, users, and other stakeholders.UML diagrams are a collection of graphical notations that can be used to represent the different aspects of a software system. There are two main types of UML diagrams: structural and behavioral.

UML stands for Unified Modeling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed and was created by the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. The UML is a very important part of developing object- oriented software and the software development process. UML uses mostly graphical notations to express the design of software projects.

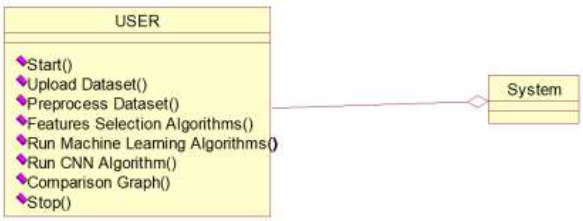
USE CASE DIAGRAM



CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

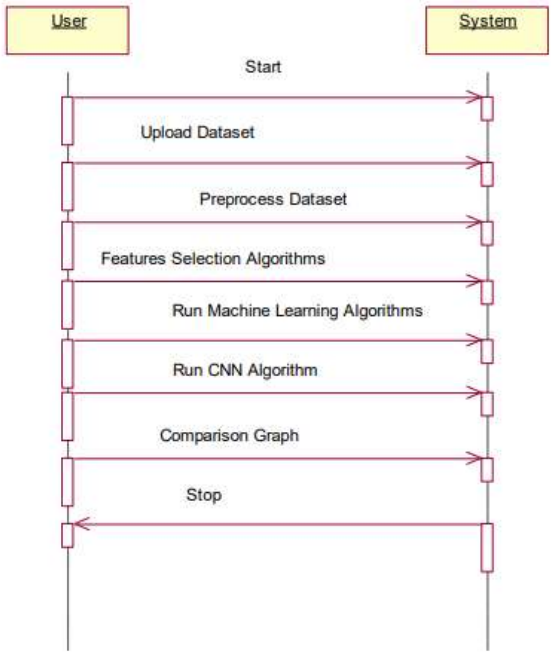
The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling.[1] The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.



In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



.

4.SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are.

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impactthat the system will have on the organization. Theamount of funds that the company can pour into the research and development of the system is limited. The expenditure must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products hadto be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. No system developed must not have a high demand on the available technical resources. This will lead to high demands for the available technical resources. This will lead to high demands being placedon the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5.MODULES

1.Numpy

Python has a strong set of data types and data structures. Yet it wasn't designed for Machine Learning per say. Enter NumPy (pronounced as Num-pee). NumPy is a data handling library, particularly one which allows us to handle large multi-dimensional arrays along with a huge collection of mathematical operations.

2.Pandas

Think of relational data, think pandas. Yes, pandas are a python library that provides flexible and expressive data structures (like data frames and series) for data manipulation. Built on top of NumPy, pandas are as fast and yet easier to use. Pandas provides capabilities to read and write data from different sources like CSVs, Excel, SQL Databases, HDFS and many more.

3.Scipy

Pronounced as Sigh-Pie, this is one of the most important

python libraries of all time. SciPy is a scientific computing library for python. It is also built on top of NumPy and is a part of the SciPy Stack.

4.Matlotlip

Another component of the SciPy stack, matplotlib is essentially a visualization library. It works seamlessly with NumPy objects (and its high-level derivatives like pandas). Matplotlib provides a MATLAB like plotting environment to prepare high-quality figures/charts for publications, notebooks, web applications and so on.

5.Scikit-learn

Designed as an extension to the SciPy library, scikit-learn has become the de-facto standard for many of the machine learning tasks. Developed as part of Google Summer of Code project ,it has now become a widely contributed open-source project with over 1000 contributors

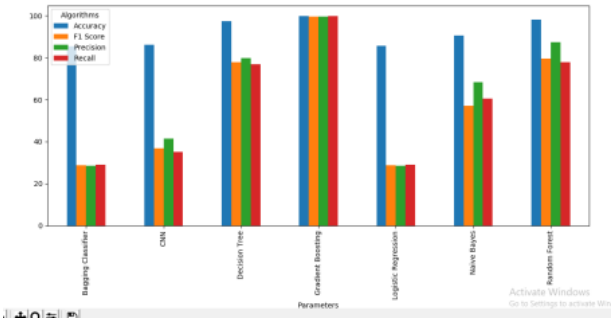
6.Seaborn

Built on top of matplotlib, seaborn is a high-level visualization library. It provides sophisticated styles straight out of the box (which would take some good amount of effort if done using matplotlib).

4. EXPERIMENTAL OUTCOME

In above screen before applying feature selection algorithm dataset contains 39 features/columns and after applying PCA feature selection we got 30 important features and dataset contains 36928 records and application using 7386 records for testing and 29542 records for training and now both train and test dataset is ready and now click on 'Run Machine Learning Algorithms' button to run all machine learning algorithms.

After 'Run CNN Algorithm' button to run CNN algorithm.



In above graph we are plotting accuracy, precision, recall and accuracy for each algorithm

5.CONCLUSION AND FUTURE SCOPE:

Conclusion:

This paper presents an experimental study for software quality prediction using machine learning methods. We used three machine learning algorithms, namely Boost, Random Forest, and Decision Tree, to predict the quality level of software modules. Our results show that all three algorithms can be used to predict software quality with a high degree of accuracy. The Boost algorithm achieved the best accuracy, followed by the Random Forest and Decision Tree algorithms.

In addition to predicting software quality, we also explored the relationship between software quality and development attributes. We found that several development attributes, such as module size, cyclomatic complexity, and lines of code, are significantly correlated with software quality. These findings suggest that

machine learning algorithms can be used to identify software modules that are at risk of being faulty.

6. 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, "Software defect 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. Bhattacharjee, "Software quality estimation using machine learning: Case-Based reasoning technique, " International Journal of Computer Applications, 2012

[6] www.isbsg.org

[7] https://goverdson.nl/

[8] H. Huygens, “Evidence-based software portfolio management: a tool description and evaluation”, 20th International Conference on Evaluation and Assessment in Software Engineering (EASE ’16), 201

[9] Arar, Ö. F., & Yilmaz, C. (2015). "A Comparative Study of Machine Learning Models for Software Defect Prediction." Information Sciences, 305, 145-161.

[10] Turan, M., & Soy Demir, E. (2017). An experimental study for software quality prediction using machine learning methods. IJRAR-International Journal of Recent Advances in Research, 2(1), 1-9.

[11] Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings." IEEE Transactions on Software Engineering, 34(4), 485-496.

[12] Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). "A systematic literature review on fault prediction performance in software engineering." IEEE Transactions on Software Engineering, 38(6), 1276-1304. Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). "A systematic literature review on fault prediction performance in software engineering." IEEE Transactions on Software Engineering, 38(6), 1276-1304.

[13] Khoshgoftaar, T. M., Allen, E. B., & Goel, N. (2009). "A comprehensive empirical study of software fault prediction performance." Journal of Systems and Software.

[14] Pan C, Lu M, Xu B, Gao H (2019) An improved CNN model for within-project software defect prediction. Appl Sci 9(10), Switzerland.

[15] Zhou T, Sun X, Xia X, Li B, Chen X (2019) Improving defect prediction with deep forest. Inf Soft Technol 114:204–216.