Review on Code Examination Proficient System in Software Engineering by Using Machine Learning Approach

Publisher: IEEE Cite This PDF

Noor Ayesha ; N G Yethiraj **All Authors**

1 Paper Citation 392 Full Text Views

Downl PDF

Abstract: The major aim of this paper is to develop code examination proficient system using machine learning algorithm in software engineering. 1. To explore the existing techniqu... **View more**

Metadata

Abstract:
The major aim of this paper is to develop code examination proficient system using machine learning algorithm in software engineering. 1. To explore the existing techniques and challenges in code review proficient system. 2. To provide effectual code review proficient system using Support Vector Machine (SVM) classification. 3. To evaluate the performance of the proposed technique using precision, accuracy, sensitivity and specificity parameters.

Published in: 2018 International Conference on Inventive Research in Computing Applications (ICIRCA)

Date of Conference: 11-12 July 2018

Date Added to IEEE *Xplore*: 03 January 2019

ISBN Information:

INSPEC Accession Number: 18358055

DOI: 10.1109/ICIRCA.2018.8597382

Publisher: IEEE

Conference Location: Coimbatore, India

SECTION I. Introduction

The increase in the demand for software systems in the daily human life has exploited the significance of the software quality in the recent years. The tolerance of unpredictable behaviour observed in the coding was noticed to decrease the run time due to the proliferation in the cost and the possibility of creating irrecoverable situations (Catal, 2011). The fortitude of the fault-prone software modules was found to be highly substantial as this process was found to identify the faulty modules which could be corrected and requires refactoring or detailed testing in order to develop a qualified software product. Consequently, the prediction of faults in the coding modules of the software were incorporated in the prediction system such that the fault proneness of future modules in the future modules could be easily detected based on the historical data concerning the faults (Sankar et aI.,2014). The failures occurring in the software were generally studied by considering numerous factors such as stochastic models which estimates the deployment of the reliability of the software's, identification of the faults in an early stage and investigating the fault prone segments to predict the faults in each segment (Erturk and Sezer, 2016).

Therefore, numerous research has been investigated to improvise the quality of the software by detecting the faults in the coding system to enhance the software development life cycle such that a high-quality software system could be developed. The significant factors which needs to be considered for improving the systems such that a reliable software could be developed were categorised as reliability, total number of failures which occurs in the coding programs (Wahono et al., 2014). In general, the Software defect prediction approaches were seen to be profitable in detecting the faults in the software as compared to software testing and reviews. Besides, the probability of predicting the fault in the coding system were predicted to be highly complex in comparison to the software reviews that where mostly used by the industrial methods (Menzies et al., 2010). Therefore, an efficient case study on the prediction model for detecting the faults in the coding system is necessary such that the system could be used for an early detection of errors in order to reduce costs, upgrade the software testing process based on defect modules and finally to improve the efficiency of the software (Hall et al, 2012). Thus, the fault detection in the coding system of the software's were found to highly significant research topic in the software engineering field (Jing et aI.,2014). Therefore, this research concentrates on investigating the code examination proficient system by utilizing a machine learning approach in the applications of software engineering.

SECTION II. Literature Review

The software fault prediction was found to address numerous issues considering the problems from diverse points such as proposing a novel method and combining methods to intensify predicting performance using attributes selection methods in order to suggest an effective metrics for the prediction. Scanniello et al. (2013) postulated a multivariate linear regression for adapting an independent variable which are found to be in correlation with faultiness in the coding system. The clusters of related classes were utilized for learning process. In Rodríguez et al., (2013), ascertained a descriptive approach for software defect prediction in the coding system based on subgroup discovery. The datasets with the respective algorithms were utilized to drive the induced rules to predict the fault. Besides, these rules was further applied to new instances classification. Dejaeger et al., (2013) examined Bayesian Network (BN) classifiers and suggested that these accessible networks with less vertex could be built to predict software defects. This research utilized 15 different Bayesian Network (BN) classifiers and further compared the with results obtained from the proposed model and other ML techniques. Therefore, the research also investigated result of the Markov blanket principle on the prediction model performance. In addition, these tests ascertained that there is no notable effect in the attribute selection technique while considering the execution of the model. Oyetoyan et al., (2013) postulated a novel method by extending object-oriented metrics such as cyclic dependencies for identifying the profile for detecting the software components. This research separated the cyclic and noncyclic metrics in order to determine the efficiency of the cyclic dependency metrics and were done at distinct class levels and package levels in prediction models. Cotroneo et al., (2013) predicted the location of Aging-Related Bugs (ARB) in order to determine the bugs in a complex system by applying software complexity metrics as predictor variables and machine learning algorithms such as NB, BN, DT, and LR with logarithmic transformation. The complexity metrics were computed after analysing the error documents of the projects that were used as dataset in the proposed work for predicting ARB-prone modules. Malhotra (2014) analysed and compared different statistical and machine learning methods such as LR, DT, ANN, Cascade Correlation Network, SVM, Group Method of Data Handling Method and Gene Expression Programming for predicting defects in diverse coding programs and to improvise the quality of the software in the developmental stage. The proposed methodology was further experimentaly validated to identify the relationship between the source code metrics and the fault susceptibility of a module for an early prediction of defects in the overall system. In Couto et al., (2014), investigated the prediction technique by considering a Granger causality test to interpret whether past modification in source code metrics values could be utilized in forecasting modifications in time series of defects. The early stage of the research considered threshold values for calculating the source code metrics and the intended threshold values and the modified classes were considered as inputs to the fault prediction system. This system was found to determine the faultiness based on the state of the class of the developed software. Czibula, et al., (2014) presented a novel classification model for detecting the faults in the software coding programs according to relational association rule mining. Besides, the research was found to consider the relation between the attributes which was analysed by applying Spearman's rank correlation coefficient to decrease the multi-dimensionality of the inputs in the data pre-operational phase. This research was further found to set of relation between the attribute values and the rule set were defined. Khosgoftaar et al., (2014) classified the software modules as fault-susceptible or not fault-susceptible based on a rule based (RB) models and case based learning (CBL).

An exhaustive literature review was performed and it was observed that an efficient technique is required to predict the defects in the coding of proposed model. To overcome the difficulties observed in the aforementioned researches many automated testing modules was identified for predicting the faults. In those modules machine learning is one of the technique. Thus, an exhaustive investigation is required based on the approach known as Stable Vector Machine (SVM) to manipulate Dimensionality reduction of heterogeneous production data such that the features of any big data can be easily extracted so that it ensures a better-quality assurance.

SECTION III. Problem Definition

In the recent past, several researches have been conducted based on developing an efficient prediction model to determine the defects in the coding technique with the primary objective to build a problem predictor model. This model was integrated with large software projects in order to determine and upgrade quality of the ultimate software model. The implementation of the static code analysis was performed in the study to increase the overall quality of the source code. Besides, by predicting the changes in the code failures, a larger focus may be provided to the endangered areas, with additional code review, in order to correct the coding errors before the testing phase. In addition, these prediction models were found to importantly increase the quality of the final software with a lower operational cost. From this study it was observed the significance of the research were correlated with the prediction of building a reliable problem prediction model. Furthermore, several software development processes were detected to be implemented in order to defect the coding errors such that an efficient prevention mechanism could be asserted for upgrading QA effectiveness in complex software by providing specific attention considering a real-life scenario. In addition, numerous classification algorithms were investigated in the application for detecting the software errors such as logistic regression, decision trees, and NN testing (Madera and Tornoń et al, 2016). However, from the above-mentioned analysis it was observed that these software defect predictions remained as an unsolved problem. Besides, it was also noticed that the correlation and benchmarking result made by the ML classifiers concerning the defect prediction in the coding system indicates an absence of the difference in performance detection and it was also ascertained the lack of an efficient classifier that performs the best for all the coding system. Furthermore, the quality assurance was found to be a significant factor in considering the development of software and it is required to reduce the risk of bugs in final product during the initial stages of the software development. However, identification of the errors or coding mistakes within the program is a challenging task as the changes in the software might have failed to execute all requirements because of failure of review of source code, static code analysis detection and testing failures like manual testing and automated.

Therefore, to overcome the above mentioned limitations, it is necessary to develop an accurate prediction system by conducting exhaustive research on the code examination proficient system.

SECTION IV. Research Methodology

The primary aim of the proposed methodology is to provide Quality Assurance (QA) and further improve the code review expert system by applying the Machine Learning (ML) algorithms. To achieve this objective, in this research, Support Vector Machine (SVM) based code review expert system is proposed. Figure 1 shows that working process of proposed rnethod.

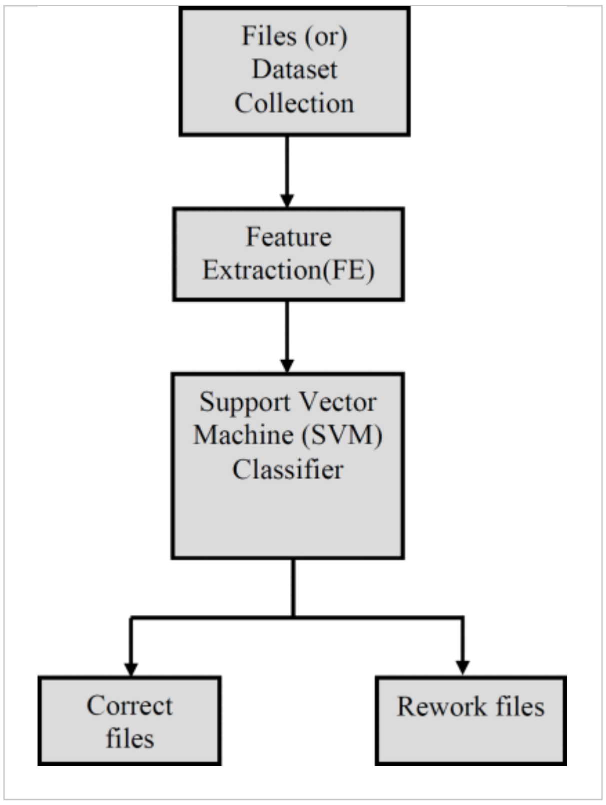
Files (or) Dataset Collection
Feature Extraction(FE)
Support Vector Machine (SVM) Classifier
Correct files
Rework files

Fig. 1
Block diagram of proposed methodology.

In data or files collection phase, there is a large number of data were collected from various data sources such as code metrics software, issue tracker, human resource database, etc. Then, five different types of features are extracted from each file in the FE phase,

1. Employee metrics: These metrics comprising the information about the author of file modification.
2. Task metrics: It consists a set of metrics related to modification request.
3. Changed file metrics: It is the characteristics related to the modified file.
4. Change quantitative metrics: It is the metrics of the file modification size.
5. Source code metrics: These metrics attained from static code testing based on the tool used.

Then, these features are trained and it is used in classification phase. Finally, the Support Vector Machine (SVM) classification is applied to categorize the correct and rework files based on the trained features.

A support vector machine (SVM) is a supervised learning algorithm which is used for binary classification. SVM is a class of machine learning algorithms referred kernel methods and are also referred to as kernel machines. A support vector machine creates an optimal hyperplane as an evaluation surface such that the margin of division between the two classes in the data is exploited. Support vectors denote a small subset of the training explanations which are used as support for the optimal location of the decision surface. Training of SVM has two stages,

1. Transform predictors to a high-dimensional feature space which is appropriate to indicate the kernel for this step.
2. Compute a quadratic optimization problem to fit an optimal hyperplane to classify the processed features into two classes.

Performance of the SVM classifier is estimated with false positive, false negative, true positive and true negative. Based on this metrics, to evaluate the overall system performance using precision, accuracy, sensitivity and specificity.

SECTION V.
Tools of Research

To evaluate the performance of the proposed research, work a flexible evaluation tool is required. The performance evaluation of proposed Support Vector Machine (SVM) based code review expert system is achieved through WEKA 3.8.1 software tool.

Weka's techniques is a data analysis tool which uses data as a single flat file, where each data exhibits its own number of attributes.

ACKNOWLEDGMENT

The proposed research solves the classification issues in code examination proficient system using machine learning algorithm. It provides more efficient results to classify the correct and rework files.

Authors

Figures

References

Citations

Keywords

Metrics