Software Defect Prediction Using Software Metrics - A survey

K.PUNITHA[1] Dr . S.CHITRA[2]
1. Asst.Prof./CSE, Bhajarang Engineering College, Tiruvallur, Chennai, TamilNadu.
Email:dr.punithak@gmail.com
2. Professor/CSE, Er. Perumal Manimekalai College of Engineering, Hosur, Krishnagiri, Tamil Nadu.

Abstract: Traditionally software metrics have been used to define the complexity of the program, to estimate programming time. Extensive research has also been carried out to predict the number of defects in a module using software metrics. If the metric values are to be used in mathematical equations designed to represent a model of the software process, metrics associated with a ratio scale may be preferred, since ratio scale data allow most mathematical operations to meaningfully apply. Work on the mechanics of implementing metrics programs. The goal of this research is to help developers identify defects based on existing software metrics using data mining techniques and thereby improve software quality which ultimately leads to reducing the software development cost in the development and maintenance phase. This research focuses in identifying defective modules and hence the scope of software that needs to be examined for defects can be prioritized. This allows the developer to run test cases in the predicted modules using test cases. The proposed methodology helps in identifying modules that require immediate attention and hence the reliability of the software can be improved faster as higher priority defects can be handled first. Our goal in this research focuses to improve the classification accuracy of the Data mining algorithm. To initiate this process we initially propose to evaluate the existing classification algorithms and based on its weakness we propose a novel Neural network algorithm with a degree of fuzziness in the hidden layer to improve the classification accuracy.
Index Terms: Software defect prediction, software defect-proneness prediction, machine learning, scheme evaluation.

1. INTRODUCTION

As our dependency on software is increasing, software quality is becoming gradually more and more important in present era. Software used almost everywhere and in every tread of life. Software consequences such as fault and failures may diminish the quality of software which leads to customer dissatisfaction [1]. A software failure is the departure of the system from its required behavior; error is the incongruity between the required and actual functionality; whereas adjudged or hypothesized cause of an error is a fault [2], which is also known as a defect (or as a bug) among software professionals [3]. Due to the increasing of complexity and the constraints under which the software is developed, it is too difficult to produce quality software. On the other hand, the software development companies cannot risk their business by shipping poor quality software [4] as it results in customer dissatisfaction. Bugs in software product cause much loss of time and money. However, learning from past experience, it would be possible to predict bugs in advance for new software products. To achieve this, we must first know which programs are more failure-prone than others. With this knowledge, we can search for properties of the program or its development process that commonly correlate with causes of bugs. Previous studies have shown that, of the overall development process 27% man hour is consumed by testing [5]. To ameliorate the testing process we can use the defect prediction models. These models can be used in defect prediction, risk analysis, effort estimation, software testability and maintainability, and reliability analysis during early phases of software development. It can also be used in business risk minimization by predicting the quality of the software in the early stages of the software development lifecycle (SDLC). This would not only help in increasing client's satisfaction but also trim down the cost of correction of defects. It has been reported in [4] that the cost of defect correction is significantly high after software testing. An additional advantage of early defect prediction is better resource planning [7] and test planning [6], [7]. Therefore, the key of developing reliable quality software within time and budget is to identify defect prone modules at an early SDLC stage by using defect prediction models. The importance of defect prediction is evident from the research work conducted in this regard.

2. PREVIOUS STUDIES

As our dependency on software is increasing, software quality is becoming gradually more and more important in present era. Software used almost everywhere and in every tread of life. Software consequences such as fault and failures may diminish the quality of software which leads to customer

dissatisfaction [1]. A software failure is the departure of the system from its required behavior; error is the incongruity between the required and actual functionality; whereas adjudged or hypothesized cause of an error is a fault [2], which is also known as a defect (or as a bug) among software professionals [3]. Due to the increasing of complexity and the constraints under which the software is developed, it is too difficult to produce quality software. On the other hand, the software development companies cannot risk their business by shipping poor quality software [4] as it results in customer dissatisfaction. Bugs in software product cause much loss of time and money. However, learning from past experience, it would be possible to predict bugs in advance for new software products. To achieve this, we must first know which programs are more failure-prone than others. With this knowledge, we can search for properties of the program or its development process that commonly correlate with causes of bugs. Previous studies have shown that, of the overall development process 27% man hour is consumed by testing [5]. To ameliorate the testing process we can use the defect prediction models. These models can be used in defect prediction, risk analysis, effort estimation, software testability and maintainability, and reliability analysis during early phases of software development. It can also be used in business risk minimization by predicting the quality of the software in the early stages of the software development lifecycle (SDLC). This would not only help in increasing client's satisfaction but also trim down the cost of correction of defects. It has been reported in [4] that the cost of defect correction is significantly high after software testing. An additional advantage of early defect prediction is better resource planning [7] and test planning [6], [7]. Therefore, the key of developing reliable quality software within time and budget is to identify defect prone modules at an early SDLC stage by using defect prediction models. The importance of defect prediction is evident from the research work conducted in this regard.

3. WORKING METHODOLOGY

The objective of this research is to build an efficient Fuzzy inference system that can learn and predict bugs in software products. Here, we have applied SVM, a supervised training algorithm for classification of data into two sets, buggy and non-buggy. Then various rules are generated inferred from the support vectors. The final set of the rules is chosen from the given set of rules using genetic algorithm optimization. The experiments were performed on Eclipse bug data on package level. Data contains the bugs reported before and after the release of product, called pre and post defects. Using the post defects as class labels for buggy and non buggy, rule base is prepared.

3.1 Performance Measurements

The performance of SVFCS is evaluated through the measures based on confusion matrix. The confusion matrix (also called a contingency table) is a measurement table which relates the predicted results with the real values of the dataset, and therefore, it makes possible to obtain measures about the hit and miss of the classifier. For a given model and set of instances, there are four possible outcomes associated with it. If the instance is defective (in fault prediction approach) and it is predicted as defective, it is counted as a true positive; if it is predicted as non-defective, it is counted as a false negative.

If the instance is non-defective and it is predicted as non-defective, it is counted as a true negative; if it is predicted as defective, it is counted as a false positive. This two-by-two confusion matrix, Fig.1, (for binary class problem) forms the basis for many common metrics. The measurements based on confusion matrix include accuracy, precision, recall, and F-measure. In literature these measures are widely used to evaluate the prediction results of a classifier. Next we explained all these measures.

Accuracy

In a binary class problem, accuracy is the ratio of the correctly classified instances to the total number of instances. It measures the hit and miss of the classifier and measures.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision

It denotes the correctness of the predicted results and measures the miss classification rate. It is defined as the ratio number of detected instances for particular class (buggy or non-buggy).

$$\Pr ecision = \frac{TP}{TP + FP}$$

Recall

The recall measures the ratio of correctly classified instances, for one class (buggy or non-buggy), to the total number of instances that really belongs to the

class. It counts the number of hits of the classifier for the class. It is also known as probability of detection and can be calculated as:

$$\text{Re}call = \frac{TP}{TP + FN}$$

However, there is a significant trade-off between recall and precision [28,29]. For instance, if model predicts only one module and it belongs to the desired class than the model's precision will be 1 but, the recall will be low if the desired class contains more than one module. In contrast, if model predicts all the defective modules correctly along with some non-defective modules incorrectly, it results in high recall and low precision. Therefore, there is a need of a combined measurement strategy such as F-measure, which efficiently combines the recall and precision in a single efficient measure.

F-measure

The F-measure combines the values of precision and recall for a classifier, for one class (faulty or not-faulty). It considers the measurements, recall, and precision, equally important and is calculated by taking their harmonic mean.

$$F - measure = \frac{2 * recall * precision}{recall + precision}$$

ROC

Receiver operating characteristics graph are well known technique for assessing the classifiers performance. It is a two dimensional graph in which abscissa represent the probability of false alarm or cost and ordinate represent the probability of detection or benefit. It can handle both discrete and continuous classifiers. Discrete classifiers output only the class level, where as continuous classifier output a curve. Each discrete classifier produces a pair of PF and PD that represent a point on ROC graph.

Definition of ROC curve contains some interesting points:

(a) Point (0, 0) denotes that it would never trigger a false alarm and also never issue a positive classification.

(b) Any point situated on the line that is drawn in between (0, 0) to (1, 1) contains no information.

(c) Point (0, 1) denotes the ideal position .But it's never achieved by any classifier. So any classifier situated close to this point, is always preferable.

4. RESULT AND DISCUSSION

Experiments were conducted by splitting the overall data set into training with 67% and testing with 33%. We repeated our experiments 10 times and randomize the input each time. This repletion is required to mitigate the impact of order effect. Results of experiments are given in Table 3 and 4 which contain the Recall (PD), Precision, Probability of false alarm (PF), Accuracy, and F-score of experiments on different data sets. We compare our results with two other model based on two well known classifiers NB and SVM. Iris and Pima Indians Diabetes data set are used for model validation. On iris data set our model gives 100% recall (Probability of detection also denoted as PD) and 0% false alarm rate (PF) with 100% accuracy where as on Pima Indians Diabetes data set it gives 68.6% and 29.5% PD and PF respectively which is better than previous known result (60% and 19%[23]) except in PF which should be low. Validation results using these two datasets suggest that we can use this model in software defect prediction.

5. CONCLUSION AND FUTURE DIRECTIONS

In this paper a novel approach, SVFCS, for fault prediction is presented. SVM, FIS, and Genetic algorithm are well known methods and are used for classification in every branch of engineering and science. In this work first time we combined the advantages of these three learners to get better prediction model. Performance of the model is checked against the open source data project where as in previous studies [10] assessment has been done on historical data sets which are publically available but could not ease the testing task in real development due to technology changes. This study need to be extended for validation purpose. Initially it works on binary classification problem we plan to generalize this model for multi class problem. We also plan to investigate the impact of support vectors on other classifier's performances.

REFERENCES

[1] Tian, J., "Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement" John Wiley & Sons, (2005).

[2] Laprie, J.C., and Kanoon, K.," Software Reliability and System Reliability, Handbook of Software Reliability Engineering". M.R. Lyu, 1,27-69, IEEE CS Press-McGraw Hill, (1996).

[3] Emam, K.,El., "The ROI from Software Quality". Auerbach Publications, Taylor and Francis Group, LLC, (2005).

[4] Khoshgoftaar, T.M., Allen, E.B., Kalaichelvan, K.S., Goel, N., "Early Quality Prediction: A Case Study in Telecommunications". 2006, IEEE Software.

[5] http://www.softwaretestingtimes.com /2010/04/ software testing-effort-estimation.htm.

[6] Khosgoftaar, T.M., Munson, J.C.,"Predicting Software Development Errors Using Software Complexity Metrics". IEEE Journal On Selected Areas In Communications1990, 8(2).

[7] Yuan, X., Khoshgoftaar, T.M., Allen, E.B., Ganesan, K., " An Application of Fuzzy Clustering to Software Quality Prediction.2000 In: Proceedings of The 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology

[8] Jiang, Y., Cukic, B., Menzies, T.,"Fault Prediction Using Early Lifecycle Data". 2007 In: Proceedings of ISSRE , TBF

[9] Basili, V., R., Briand, L., C., Melo, W., L, " A validation of object-oriented design metrics as quality indicators".1996, IEEE Trans. on Software Engineering. 22, 751-761.

[10] Bharavi Mishra and K.K. Shukla " Defect Prediction for Object Oriented Software using Support Vector based Fuzzy Classification Model" International Journal of Computer Applications, Vol. 60, No.15, 2012.