Stochastic Modelling and Computational Sciences

DEVELOPING AN OBJECT-ORIENTED ADAPTIVE FAULT PREDICTION FRAMEWORK FOR COST EVALUATION IN BIOMETRIC SOFTWARE

Waseem Ahmad[1], Deepa. P. Sukumaran[2], Anto Pratheesh Jose T[3], Anand Mohanrao Magar[4], Ruch Ohri[5] and Divya Sharma[6]

[1,2]Research Scholar, Department of Computer Science, Shri Venkateshwara University, Gajraula, Up, India
[3]Research Scholar, Department of Electronics & Communication Engineering, Shri Venkateshwara University, Gajraula, Up, India
[4]Research Scholar, Department of Computer Science & Engineering, Shri Venkateshwara University, Gajraula, Up, India
[5]Research Scholar, Department of Computer Science & Engineering, Nims Institute of Engineering and Technology (Niet), Nims University, Rajasthan, Jaipur, Rajasthan, India
[6]Research Scholar, Department of Computer Science And Engineering, Nims Institute of Engineering and Technology (Niet), Nims University, Rajasthan, Jaipur, Rajasthan, India

ABSTRACT

Fault prediction is necessary in software development life cycle in order to reduce the probable software failure and is carried out mostly during initial planning to identify fault-prone modules. Many fault-prediction techniques have been proposed and evaluated for their performance using various performance criteria. However, due to the lack of compiling their performances in proper perspective, one significant issue about the viability of these techniques has not been adequately addressed. In the proposed work, an adaptive cost evaluation framework is proposed that incorporates cost drivers for various fault removal phases, and performs a cost-benefit analysis for the misclassification of faults. Accordingly, the proposed work focuses on investigating two important and related research questions regarding the viability of fault prediction. First, for a given project, does the developer feel that the fault prediction results useful? In case of an affirmative answer, then it is desirable to investigate as to how to choose a fault prediction technique for an overall improved performance in terms of cost effectiveness.

Keywords : Object Oriented, Software Metrics, Regression, Fault Prediction, Biometric

INTRODUCTION

Fault prediction not only gives an insight to the need for increased quality of monitoring during software development but also provides necessary tips to undertake suitable verification and validation approaches that eventually lead to improvement of efficiency and effectiveness of fault prediction. Software fault prediction is helpful in deciding the amount of effort needed for software development. Effectiveness of a fault prediction is studied by applying a part of previously known data related to faults and predicting its performance against other part of the fault data. Several researchers have worked on building prediction models for software fault prediction but less emphasis has been given on the study of effectiveness of fault prediction. However, very less significant work has been done on feasibility of fault prediction approach. Software defect can be requirement defect, design defect, code defect, test case defect and other product defect. These metrics help to verify the quality attributes of software such as effort and fault proneness. The usefulness of these metrics lies in their ability to forecast the quality of the software. In literature it is observed that, a good number of approaches have been studied and evaluated on software products to determine best suitable approach for fault prediction based on certain performance criteria (precision, recall, accuracy etc.).

LITERATURE SURVEY

Present day software development is mostly desired to be based on Object-Oriented (OO) parametric are used for measure the performance. Basili [1] experimentally analyzed the impact of CK metric suite in fault prediction. In his study, he correlated WMC, DIT, NOC, RFC and CBO with defects for eight academic projects. He experimentally analyzed the impact of CK metric suite in fault prediction. Briand *[*2] found out the relationship between fault and the metrics using uni-variate and multivariate logistic regression models. Tang [3] investigated

the dependency between CK metric suite and the Object-Oriented system faults. Emam [4] conducted empirical validation on Java application and found that export coupling has great influence with faults. He Found that size confound the effect of all metrics on fault proneness for large telecommunication application. Khoshgoftaar [5], Hochman [6] conducted experimental analysis on telecommunication model and found that artificial neural network (ANN) model is give accurately output than another discriminant model. In their approaches, nine software metrics were used for modules developed in procedural paradigm. Since then, ANN models have taken a rise in their usage for prediction modeling. Hence, in this study, different ANN models are used for fault prediction of embedded software. Mende [7] introduced a performance measure (Popt) and compared prediction model with an optimal model. Mende [8] proposed two strategies namely AD (effortaware binary prediction) and DD (effortaware prediction based on defect density).

The proposed framework is used to investigate the usefulness of various fault-prediction techniques. The investigation consisted of performance evaluation of five major fault-prediction techniques i.e., liner regression, polynomial regression, logistic regression, Naive Bayes and SVM. From the obtained results, it is observed that applications of fault prediction models are useful for the projects with percentage of faulty modules less than a certain threshold.

RESEARCH METHODOLOGY

The following sub-sections give the basic definitions of the performance parameters used for fault prediction.

Table 1: *Confusion Matrix to classify a class as Yes faulty and not faulty*

	Not-Faulty	**Yes Faulty**
Not-Faulty	Yes Negative (YN)	Not Positive (NP)
Yes Faulty	Not Negative (NN)	Yes Positive (YP)

The confusion matrix is categories into four categories:

1. Yes positives (YP) are the number of modules correctly classified as faulty modules.
2. Not positives (NP) refer to not-faulty classes incorrectly labelled as faulty classes.
3. Yes negatives (YN) correspond to not-faulty modules correctly classified as such.
4. Finally, Not negatives (NN) refer to faulty classes incorrectly classified as not-faulty classes.

According to authors in [1], following are the performance parameter used to measures the classification techniques.

Precision: It is used to measure the degree to which the repeated measurements under unchanged conditions show the same results.

$$Precision = \frac{TP}{FP+TP} \quad \text{.....(1)}$$

Recall: It indicates the how many of the relevant things which are to be identified. It is represented as:

$$Recall = \frac{TP}{FN+TP} \text{.....(2)}$$

F-Measure: These are used to join the precision and recall numeric value to produce one score, which is defined as the harmonic mean of the recall and precision. It is computed as:

$$F - Measure = \frac{2*Recall*Precision}{Recall+Precision} \text{... .(3)}$$

Specificity: Specificity indicates how effectively a classifier identifies the negative labels. It may be expressed as:

$$Specificity = \frac{TN}{FP+TN} \text{.....(4)}$$

Accuracy: Accuracy measure is the proportion of predicted fault prone modules that are inspected out of all modules. It is defined as:

$$Accuracy = \frac{TN+TP}{TP+TN+FP+FN} \ldots\ldots\ldots\ldots (5)$$

EXPERIMENTAL STUDY

In this section, the experimental study done to find the effectiveness of fault prediction techniques for the cost-based evaluation framework is presented. In this study, five techniques such as linear regression, polynomial regression, logistic regression, naive bayes, and support vector machine are used to find the classification accuracy. These five techniques are employed on Ellipse JDT core from PROMISE data repository.

Software fault prediction is helpful in deciding the amount of effort needed for software development. In literature it is observed that, a good number of approaches have been studied and evaluated on software products to determine best suitable approach for fault prediction based on certain performance criteria (precision, recall, accuracy etc.). However very less significant work has been done on feasibility of fault prediction approach. In this study, a cost evaluation framework has been proposed which performs cost-based analysis for misclassification of faults. Accordingly, this study focuses on investigating two important and related research questions regarding the viability of fault prediction. First, for a given project, does the developer feel that the fault prediction results useful? In case of an affirmative answer, then it is desirable to investigate as to how to choose a fault prediction technique for an overall improved performance in terms of cost effectiveness. The proposed framework is used to investigate the usefulness of various fault-prediction techniques. The investigation consisted of performance evaluation of five major fault-prediction techniques i.e., liner regression, polynomial regression, logistic regression, Naive Bayes and SVM. From the obtained results, it is observed that applications of fault prediction models are useful for the projects with percentage of faulty modules less than a certain threshold.

A. Fault Data

To perform statistical analysis, bugs were collected from Promise data repository [9]. Table 2 shows the distribution of bugs based on the number of occurrence (in terms of percentage of class containing number of bugs.

Table 2: *Distribution of bugs from PROMISE [9]*

No. of Classes	% of Bugs	Number of Associated Bugs
791	79.33	0
138	13.84	1
31	3.1	2
15	1.5	3
8	0.8	4
2	0.2	5
4	0.4	6
3	0.3	7
3	0.3	8
2	0.2	9

Proposed case study shown in Figure 1 contains 997 number of different classes in which 79.33% of classes contain zero bugs i.e., out of 997 classes: 791 classes contain zero bugs, 13.84% of classes contain at least one bug, 3.10% of classes contain a minimum of two bugs, 1.50% of classes contain three bugs, 0.80% classes contain four bugs, 0.20% of classes contain five and nine bugs, 0.40% classes contain six bugs, 0.30% of classes contain seven and eight bugs.

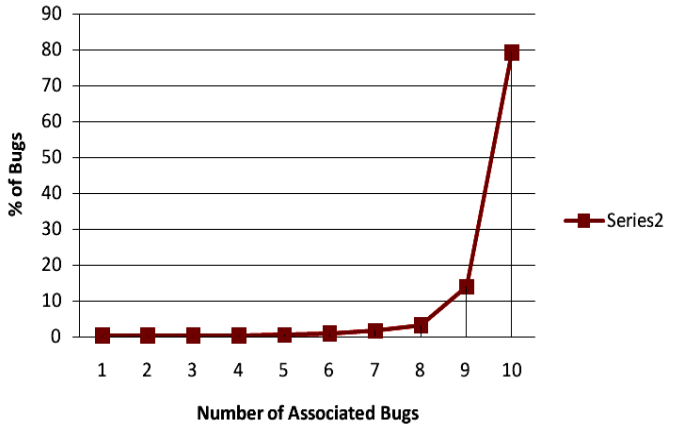

Figure 1 *Distribution of bugs in %age*

B. Cost Analysis Model

$$Ecost = C_i + C_u * (FP + TP) + \delta_s * C_s * (FN + (1 - \delta_u) * TP) + (1 - \delta_s) * C_f * (FN + (1 - \delta_u) * TP) \quad \text{.... (6)}$$

$$Tcost = M_p * C_u * TC + \delta_s * C_s * (1 - \delta_u) * FC + (1 - \delta_s) * C_f * (1 - \delta_u) * F \quad \text{.... (7)}$$

$$NEcost = \frac{Ecost}{Tcost} = \begin{cases} <1 & Fault\ Prediction\ is\ Useful \\ \geq 1 & Perform\ Testing \end{cases} \quad \text{.... (8)}$$

Ecost represents for Estimated fault removal cost of the software when fault prediction is performed.

TCost is the Estimated fault removal cost of the software without using fault prediction approach. Necost represents the Normalized Estimated fault removal cost of the software when fault prediction is utilized.

Linear Regression: Linear regression is one of the most commonly used statistical technique [12]. It is used to find the linear (i.e., straight-line) relationship between variables. The Uni-variate linear regression is expressed a given in below equation 9.

$$Y=\beta_1 X+\beta_0.$$

Where Y is a dependent variable and X denotes an independent variable. 0,1 are the constant and coefficient values respectively.

Polynomial Regression: . Polynomial models are used in problems where the researcher knows that curvilinear effects are present in the true response function. The Uni-variate polynomial regression

$$Y=\beta_0+\beta_1 X+\beta_2 X^2+\ldots. +\beta_n X^n$$

analysis is expressed a given equ-10 as:

where Y is dependent variable, X is independent variable which are the constant and coefficient values respectively.

Logistics Regression: It is a type of regression analysis used for predicting the outcome of dependent variable based on one or more independent variables The logistic regression model is based on the eq.11:

$$logit[\pi(x)] = logit[\pi(x)] = \beta_0+\beta_1 X_1+\beta_2 X_2+\ldots..+\beta_m X_m$$

where xi represent the independent variable and logit[(x)] represent the dependent variable

Naïve Bayes: Naive Bayes classifier assigns the given object x to class $e^* = \text{argmax}_d P(d|x)$ by using Bayes′ rule given below:

where P(d), is the prior probability of a parameter c before having seen the data. P(d|x) is called the likelihood. It is the probability of the data x and defined as

Support Vector Model: SVM is one of the supervised machine learning models which is generally used for classification and regression analysis. SVM model analyses data and recognizes the patterns involved in the data set. The general form of SVM function is expressed as given in eq.12.

$$Y' = w * \phi(x) + b$$

where $\phi(x)$ is non-linear transform. The main goal of this study is to calculate the value of w and b, so the value of Y' can be found by minimization of regression risk.

RESULT AND ANALYSIS

In this experiment, the values tabulated in Table 3 have been used in design of cost evaluation model. δu and δs show the fault identification efficiency of unit testing and system testing, respectively. The values of δu and δs have been collected from the survey report "Software Quality in 2010" [11]. Mp shows the fraction of modules unit tested, obtained from the paper of Wilde [10]. The objective is to provide the bench marks to approximate the overall fault removal cost. This is clear from the proposed cost evaluation model that if a technique is having high false negatives and/or high false positive, then it results in higher fault removal cost. When this approximated cost exceeds the unit testing cost (Tcost), it is cost effective to test all the modules at unit level instead of using fault prediction.

Table 3 and Figure 2, lists the values of obtained performance parameters for Ellipse JDT core data

- Logistics regression technique obtained promising classification rate when compared to other four techniques.
- It can be concluded that NEcost was less than 1 for the jdt data set for all the five techniques. Logistic regression incurred negligibly less NEcost in comparison to other techniques.

Figure 2: Snapshot 5 Support Vector Model

Table 3: Result of experiment

Technique	Specification	Recall	Precision	F-Measure	Accuracy	NEcost
Linear Regression	0.9210	0.5161	0.6	0.5549	0.8455	0.8943
Polynomial Regression	0.9627	0.4063	0.7542	0.5281	0.8405	0.8879
Logistics Regression	0.9722	0.3414	0.7602	0.4713	0.8425	0.8823
Naives Bayes	0.9721	0.3269	0.7555	0.4563	0.8375	0.8871
SVM	1	0.3464	1	0.5146	0.8505	0.8886

In this section, the relationship between value of metrics and the fault found in a class is determined. The comparative study involves using six CK metrics as input nodes and the output is the achieved fault prediction rate. Figure 4.1 shows the flow chart for the proposed cost-based evaluation framework.

Table 3 and Figure 3, show the classification matrix for jdt data set for the applied techniques such as linear regression technique, polynomial regression technique, logistic regression technique, navies bayes classification and SVM method.

- ***Linear Regression Technique:*** it can be observed that in case of linear regression technique, total number of 843 (747+96) classes were classified correctly with 84.55% accuracy rate. .
- ***Polynomial Regression Technique*****:** it can be observed that in case of polynomial regression technique, total number of 838 (749+89) classes were classified correctly with 84.05% accuracy rate.
- ***Logistic Regression Technique*****:** it can be observed that in case of logistic regression technique, total number of 840 (770+70) classes were classified correctly with 84.25% accuracy rate. .
- ***Naive Bayes Technique*****:** it can be observed that in case of navies Bayes technique, total number of 835 (767+68) classes were classified correctly with 83.75% accuracy rate.
- ***SVM Technique:*** it can be observed that in case of SVM technique, total number of 848 (769+79) classes were classified correctly with 85.06% accuracy rate.

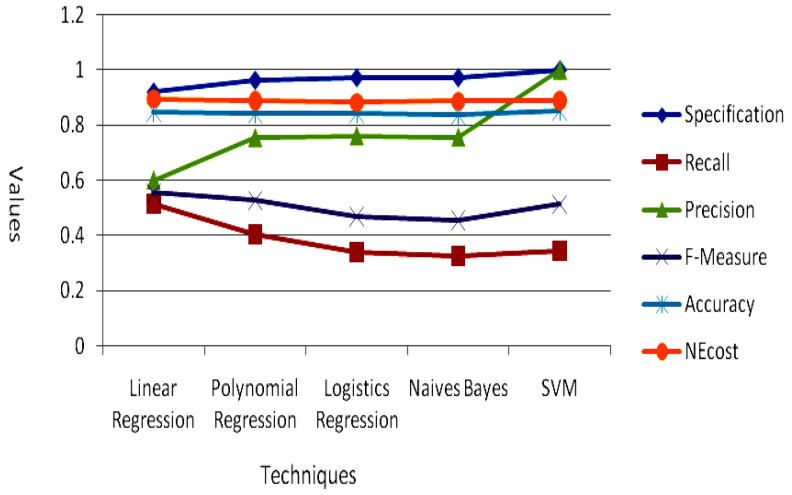

Figure 3: Graphical Representation of experimental results

CONCLUSION AND FUTURE WORK

Software quality assurance process focuses on the identification and removal of faults quickly from the artifacts that are generated and subsequently used in the development of software. Software fault prediction is one of the different strategies, which are conducted rapturously during the very early stage of software development life cycle (SDLC). Fault prediction information not only points to the need for increased quality during the development but also provides information to understand suitable verification and validation activities in order to improve the effectiveness. The proposed framework is used to investigate the usefulness of various fault prediction techniques. From the obtained results, it is observed that fault prediction approach is useful for the projects with percentage of faulty modules less than a certain threshold. It was observed that there was no single technique that could provide the best results in all cases. However, for different critical applications, where ignoring faults can be crucial, using fault prediction may not be effective if it has high false negatives. Otherwise, it may result in a poor-quality outcome. Further, work can be carried out on another quality parameter such as: software maintainability. Software Maintainability is generally measured as a change effort. Change effort may be taken as either the average effort needed to make a change to a class, or the total effort used on modifying a class.

REFERENCES

[22] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of Object-Oriented design metrics as quality indicators," IEEE Transactions on Software Engineering, vol. 22, pp. 751–761, 2016.

[23] L. C. Briand, J. W¨ust, J. W. Daly, and D. V. Porter, "Exploring the relationships between design measures and software quality in Object-Oriented systems," The Journal of Systems and Software, vol. 51, pp. 245–273, 2020.

[24] M.-H. Tang, M.-H. Kao, and M.-H. Chen, "An empirical study on objectoriented metrics," in Software Metrics Symposium, 1999. Proceedings. Sixth International, pp. 242–249, IEEE, 1999.

[25] K. El Emam, S. Benlarbi, N. Goel, and S. N. Rai, "The confounding effect of class size on the validity of object-oriented metrics," IEEE Transactions on Software Engineering, vol. 27, no. 7, pp. 630–650, 2021.

[26] H. J. A. S. Khoshgoftaaar T.M, Allen E.B, "Application of neural networks to software quality modeling of a very large telecommunications systems," IEEE Transactions on Neural Networks, vol. 8, no. 4, pp. 902–909, 2019.

[27] F.B. Abreu and R. Carapuca, "Candidate Metrics for ObjectOriented Software within a Taxonomy Framework", System and Software, vol. 26, no. l, pp. 87-96, 2022.

[28] M. T and K. R, "Revisiting the evaluation of defect prediction models," in Proceedings of the 5th International Conference on Predictor Models in Software Engineering (PROMISE), pp. 1–10, 2019.

[29] M. T and K. R, "Effort-aware defect prediction models," in Proceedings of 14th IEEE European Conference on Software Maintenance and Re-engineering (CSMR), pp. 107–116, 2010.

[30] Michael Fagan, "A history of software inspections, Software pioneers: contributions to software engineering", Springer-Verlag New York, Inc., New York, NY, 2012

[31] R. Huitt and N. Wilde, "Maintenance support for object-oriented programs," IEEE Transactions on Software Engineering, vol. 18, no. 12, pp. 1038–1044,2012.

[32] Y. Zhou and H. Leung, "Predicting object-oriented software maintainability using multivariate adaptive regression splines," Journal of Systems and Software, vol. 80, no. 8, pp. 1349–1361, 2007.

[33] Michael Fagan, "A history of software inspections, Software pioneers: contributions to software engineering", Springer-Verlag New York, Inc., New York, NY, 2022