RESEARCHERID
THOMSON REUTERS

[Gandhi * *et al.*, 7(1): January, 2018]
IC™ Value: 3.00

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A REVIEW ON SOFTWARE PRONE DETECTION AND ITS PREVENTION TECHNIQUES

Laxmi Dewangan[*1] & Prof. Anish Lazrus[2]

[*1&2]Shri Shankaracharya Group of Institutions, Dept. of Information & Technology, Junwani Bhilai, Chhattisgarh, India

DOI: 10.5281/zenodo.1161695

ABSTRACT

The need of distributed and complex business applications in big business requests error free and quality application frameworks. This makes it critical in programming improvement to create quality and fault free programming. It is likewise critical to outline dependable and simple to keep up as it includes a great deal of human endeavors, cost and time amid programming life cycle. A software advancement process performs different exercises to limit the faults, for example, fault prediction, defect localization, prevention and amendment. This paper shows a study on current practices for programming fault location and counteractive action systems in the product advancement. It additionally talks about the focal points and impediments of these systems which identifies with the quality item improvement and support.

KEYWORDS: Software Fault Prediction, Prone Detection, SVM, Prediction Techniques.

I. INTRODUCTION

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. 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 [1,2].

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 [3].

So as to evaluate the unwavering quality of a class, a few conventional strategies are accessible in the writing. Be that as it may, less significance has been given on utilizing machine learning techniques [4]. AI techniques, a subset of machine learning strategies have the capacity of PC, programming and firmware to gauge the properties of a class that individuals perceive as insightful conduct. These techniques can estimated the non-straight capacity with more accuracy. Consequently they can be connected for quality estimation keeping in mind the end goal to accomplish better exactness [5,6].

TABLE I. Confusion Matrix to Classify the Class

	Non-Faulty	Faulty
Non-Faulty	True Negative (TN)	False Positive (FP)
Faulty	False Negative (FN)	True Positive (TP)

The confusion matrix are classes into four classifications:

- True positives (TP) are the quantity of modules accurately classified as flawed modules.
- False positives (FP) allude to not-flawed classes inaccurately marked as broken classes.

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

- True negatives (TN) compare to not-broken modules effectively classified in that capacity.
- Finally, false negatives (FN) allude to flawed classes inaccurately classified as not-defective classes.

II. PARAMETERS FOR PERFORMANCE COMPARISONS OF SOFTWARE PRONENESS EVAULUATION

The accompanying sub-areas give the essential meanings of the execution parameters utilized for prone prediction [7].

A. Precision

It is utilized to quantify how much the repeated estimations under unaltered conditions demonstrate similar outcomes.

$$Precision = \frac{TP}{FP + TP}$$

B. Recall

It shows the what number of the applicable item that are to be recognized. it is represented as:

$$Recall = \frac{TP}{FN + TP}$$

C. F-Measure

F-Measure join the exactness and recall numeric value to give a solitary score, which is characterized as the harmonic mean of the recall and accuracy. F-Measure is communicated as:

$$F - Measure = \frac{2 * Recall * Precision}{Recall + Precision}$$

D. Specificity

Specificity concentrate on how adequately a classifier recognizes the negative labels. It is characterized as:

$$Specificity = \frac{TN}{FP + TN}$$

E. Accuracy

Accuracy measure is the extent of anticipated fault-inclined modules that are investigated out of all modules. It is characterized as:

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

Fault prediction is useful in choosing the measure of exertion required for programming improvement. A great number of methodologies have been considered and assessed on programming items to decide best reasonable approach for fault prediction in light of certain execution criteria (Accuracy, recall, precision and so forth.) [8, 9]. However less critical work has been done on achievability of fault prediction approach. In this investigation, a cost assessment structure has been proposed which performs cost based examination for misclassification of faults [10].

III. FAULT PREDICTION TECHNIQUES

A. Linear Regression Models

Linear regression is the regularly utilized statistical system. It is utilized to locate the direct (i.e., straight-line) connection between factors.

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

The Univariate straight relapse is spoken to as:

Y = ß1X + ß0

Where Y represent the dependent variable and X represent the independent variable. ß0, ß1 are the constant and coefficient values respectively.

B. *Polynomial Regression Models*

Polynomial regression is the ordinarily utilized measurable strategy. Polynomial models are valuable in circumstances where the examiner realizes that curvilinear impacts are available in the genuine reaction work. Polynomial models are additionally helpful as approximating capacities to obscure and conceivable extremely complex nonlinear relationship.

C. Logistic regression model

Logistic regression is the ordinarily utilized measurable method. Which is a sort of regression examination utilized for anticipating the result of ward variable in light of at least one free factors. A needy variable can take just two esteems. So the needy variable of a class containing bugs is separated into two gatherings, one gathering containing zero bugs and the other having no less than one bug. Strategic relapse display is utilized to build a prediction demonstrate for the fault inclination of classes. In this strategy, measurements are utilized as a part of blend.

D. *Naive Bayes model*

Naive Bayes is one of the approach for outline the classifier. It is a straightforward probabilistic classifier which depend on applying Bayes' hypothesis with solid freedom suspicions. A more illustrative term for the hidden likelihood model would be "autonomous element display".

E. Support Vector Machine model

SVM is one of the administered machine learning model which is for the most part utilized for classification and relapse investigation. SVM show examinations information and perceives the examples associated with the informational index. SVM demonstrate goes about as a non-probabilistic twofold straight classifier by sorting input information into same classification or the other.

IV. LITERATURE SURVEY

Lee et al. [11], Author directs an examination of basic points around there, including techniques for assessing the viability of fault-inclination prediction models, issues of concerns when fabricating the prediction models, and also discoveries shared by the scholastic group. The proposed technique can't configuration to take skewed dataset. Thus require assist improvement.

Arisholm et al. [12] proposed cost viability (CE) by requesting the modules by their anticipated imperfection thickness and after that plotting the quantity of real faults found against the aggregate sum of code investigated beginning with the modules with the most astounding anticipated deformity thickness.

Russo [13] presented the TTV (Train, Test, and Validate) calculation went for formalizing the model preparing and prediction assessment process. TTV contains three stages: preparing a prediction model to locate the best parameters of the model; utilizing the prepared model on new information to figure the prediction execution; and approving the model on encourage new information for prediction.

The proposed TTV intends to formalize the Lessmann et al. [14] proposal that the appraisal and determination of a fault-inclination prediction model ought not be founded on classification exactness alone but instead ought to be contained a few extra criteria like computational proficiency, usability, and fathomability.
D. Rodrguez et al. [15], Author propose EDER-SD (Evolutionary Decision Rules for Subgroup Discovery), a this calculation in view of developmental calculation that instigates rules depicting just fault-inclined modules. EDER-SD has the benefit of working with nonstop factors as the states of the guidelines are characterized utilizing interims.

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

Martin Shepperd et al. [16], Author contemplated on the openly accessible NASA datasets have been broadly utilized as a component of this exploration to classify programming modules into deformity inclined and not desert inclined classifications. In such manner, the Promise Data Repository 2 has served an essential part in making programming building informational indexes freely accessible. For instance, there are 96 programming deformity datasets accessible.

Ezgi Erturk et al. [17], Author proposed another technique Adaptive Neuron Fuzzy Inference System (ANFIS) for the product fault prediction. Information are gathered from the PROMISE Software Engineering Repository, and McCabe measurements are chosen since they completely address the programming exertion. The outcomes accomplished were 0.7795, 0.8685, and 0.8573 for the SVM, ANN and ANFIS strategies, separately.

Martin Shepperd et al. [18], Author displayed a novel benchmark structure for programming imperfection prediction. In the system includes both assessment and prediction. In the assessment arrange, distinctive learning plans are assessed by that plan chose. At that point, in the prediction arrange, the best learning plan is utilized to manufacture an indicator with every single authentic datum and the indicator is at long last used to anticipate imperfection on the new information

David Gray [19], in this paper the principle concentrate is on classification examination as opposed to classification execution, it was chosen to classify the preparation information instead of having some type of analyzer set. It includes a manual investigation of the predictions made by Support vector machine classifiers utilizing information from the NASA Metrics Data Program storehouse.

Surndha Naidu et al. [20], in this paper concentrated on finding the aggregate number of imperfections with a specific end goal to decrease the time and cost. Here for imperfection classification utilized ID3 (Iterative Dichotomiser calculation. ID3 is a calculation imagined by Ross Quinlan used to create a choice tree from a dataset. The deformities were classified in light of the five characteristic esteems, for example, Volume, Program length, Difficulty, Effort and Time Estimator

TABLE I. Comparisons of various techniques and method used in present system

SNO	Author	Dataset Used	Method	Finding
1	Lee et al.	OOPS Metrics	Feature selection and reduction	Reduces the risk of software faults manifesting during operation, techniques are needed to identify code which has the potential to cause problems early on so that more effort can be spent on testing to prevent such problems from occurring.
2	Arisholm et al.	OOPS Metrics	C4.5 decision trees	Author build prediction based on the probability for a class to contain at least a fault based on a number of variables characterizing the class source code, amount of change in the last release, change and fault history over past releases, and developer's experience and number.
3	Russo et al.	OOPS Metrics	Train, Test and Validation Algorithm	The framework is expressed as three consecutive algorithms and want to control two crucial issues that can hamper a comparison across studies: compute and report accuracy of fitting of a model and determine the set of parameters needed to describe the solution.
4	Lessmann et al.	OOPS Metrics	Multiple Learning Techniques	Results indicate that the importance of the particular classification algorithm may be less than previously assumed since no significant performance differences could be detected among the top 17 classifiers.
5	D. Rodrguez et al.	OOPS Metrics	Feature Extraction and Reduction	Object-oriented metrics (49%) were used nearly twice as often compared to traditional source code metrics (27%) or process metrics (24%). There are significant differences between the metrics used in fault prediction performance.

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

6	Martin Shepperd et al.	OOPS Metrics	Machine Learning Approach	Author finds important differences between the two versions of the datasets, implausible values in one dataset and generally insufficient detail documented on dataset pre-processing.
7	Ezgi Erturk et al.	OOPS Metrics	Support Vector Machine	ROC-AUC is used as a performance measure. The results achieved were 0.7795, 0.8685, and 0.8573 for the SVM, ANN and ANFIS methods, respectively.
8	Martin Shepperd et al.	OOPS Metrics	ANOVA model	Author's find that the choice of classifier has little impact upon performance (1.3 percent) and in contrast the major (31 percent) explanatory factor.
9	David Gray et al.	OOPS Metrics	Support Vector Machine	The purpose of method was to gain insight into how the classifiers were separating the training data.
10	Surndha Naidu et al.	OOPS Metrics	ID3 or C4.5	The defects were classified based on the five attribute values such as Volume, Program length, Difficulty, Effort and Time Estimator. After the classification of defects they were measured using the pattern mining technique. The quality was assured using the quality metrics such as defect density and the accuracy was assured by sensitivity and specificity.

V. CONCLUSION

Today there is an inborn requirement for programming unwavering quality is getting expanded consideration nowadays and exceedingly fault tolerant framework. In this overview paper, examine on fault recognition component, and additionally fault counteractive action system in connection to the current pattern of the most recent advancements have been talked about.

There are many flaws in existing system which are further need to be improved. The flaws such as while considering skewed and missing dataset and also running automatic fault prediction technique via some robots which are also not that much accurate.

VI. REFERENCES

[1] F. B. E. Abreu and R. Carapuca, “Object-Oriented software engineering: Measuring and controlling the development process,” in Proceedings of the 4th International Conference on Software Quality, vol. 186, 1994.
[2] J. Bansiya and C. G. Davis, “A hierarchical model for Object-Oriented design quality assessment,” ACM Transactions on Programming Languages and Systems. vol. 128, pp. 4–17, August 2002.
[3] B. K. Kang and J. M. Bieman, “Cohesion and reuse in an Object-Oriented system,” in Proceedings of the ACM SIGSOFT Symposium on software reuseability, pp. 259–262, Seattle, March 1995.
[4] 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, May 2000.
[5] L. Etzkorn, J. Bansiya, and C. Davis, “Design and code complexity metrics for Object-Oriented classes,” Object-Oriented Programming, vol. 12, no. 10, pp. 35–40, 1999.
[6] M. Halstead, Elements of Software Sciencel. New York, USA: Elsevier Science, 1977.
[7] B. Henderson-Sellers, Software Metrics. UK: Prentice-Hall, 1996.
[8] W. Li and S. Henry, “Maintenance metrics for the Object-Oriented paradigm,” in Proceedings of First International Software Metrics Symposium, pp. 52–60, 1993.
[9] T. J. McCabe, “A complexity measure,” IEEE Transactions on Software Engineering, vol. 2, pp. 308–320, December 1976.
[10] D. P. Tegarden, S. D. Sheetz, and D. E. Monarchi, “A software complexity model of Object-Oriented systems,” Decision Support Systems, vol. 13, no. 3, pp. 241–262, 1995.
[11] S. Y. Lee, D. Li and Y. Li, "An Investigation of Essential Topics on Software Fault-Proneness Prediction," 2016 International Symposium on System and Software Reliability (ISSSR), Shanghai, 2016, pp. 37-46.

ISSN: 2277-9655
Impact Factor: 5.164
CODEN: IJESS7

[12] E. Arisholm, L. C. Briand, and M. Fuglerud, "Data Mining Techniques for Building Fault-Proneness Models in Telecom Java Software," in Proceedings of the 18th IEEE International Symposium on Software Reliability Engineering, Trollhättan, Sweden, pp. 215-224, November 2007
[13] B. Russo, "A Proposed Method to Evaluate and Compare Fault Predictions Across Studies," in Proceedings of the 10th International Conference on Predictive Models in Software Engineering, Turin, Italy, pp. 2-11, September 2014
[14] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," IEEE Transactions on Software Engineering, vol. 34, no. 4, pp. 485-496, August 2008
[15] D. Radjenovic, M. Heri_cko, R. Torkar, and A. Zivkovi_c, Software fault prediction metrics: A systematic literature review," Information and Software Technology, vol. 55, no. 8, pp. 1397-1418, 2013.
[16] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: some comments on the nasa software defect datasets," Software Engineering, IEEE Transactions on, vol. 39, no. 9, pp. 1208 -1215, 2013.
[17] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," Expert Systems with Applications, 2014.
[18] M. Shepperd, D. Bowes, and T. Hall, "Researcher bias: The use of machine learning in software defect prediction," Software Engineering, IEEE Transactions on, vol. 40, no. 6, pp. 603-616, 2014.
[19] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "Software defect prediction using static code metrics underestimates defect-proneness," in Neural Networks (IJCNN), The 2010 International Joint Conference on, pp. 1-7, IEEE, 2010.
[20] Naidu, M. Surendra, and N. GEETHANJALI. "Classification of Defects in Software using Decision Tree Algorithm." International Journal of Engineering Science and Technology (IJEST) 5.06 (2013).", Printice Hall india,2014.

CITE AN ARTICLE

Dewangan, L., & Lazrus, A., Prof. (n.d.). A REVIEW ON SOFTWARE PRONE DETECTION AND ITS PREVENTION TECHNIQUES. *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY, 7*(1), 598-603.