JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4790-4795
https://publishoa.com
ISSN: 1309-3452

Software Defect Prediction System Using Rayleigh Distribution

Madhumitha R, RagaviIswariya M, Padmavathi S
Sri Krishna College of Technology

Received: 2022 March 15; **Revised:** 2022 April 20; **Accepted:** 2022 May 10

Abstract
Software development organizations are devoting ideas, time, and money to Improve Software Process since it helps to improve product quality while also lowering process time and cost. Existing methodologies and techniques consumes time and are expensive, and their primary objective is on big software companies. As a result, we've introduced Software Process Improvement for resolving these issues.

Keywords: Software process improvement, Defects, Software development, Defect prediction, Rayleigh distribution

1. Introduction

Software system is a target process in software development that comprises of partly organized phases that are followed to achieve an aim or goal. Programming, evaluating, scheduling, and packaging are all required in the development of a software process. These processes may be improved for higher quality, reduced time, price, and product delivery, and the activity that improves these processes is called Software Process Improvement. The number of defects is necessary for predicting the software quality but the accurate estimation of defect can be a difficult. Few techniques generally assume that the faults found are a sample of the all existing faults, which results in inaccurate estimates. Other techniques provide little information in addition to the number of faults already found. Hence we describe a simple procedure for estimating the total number of actual defects.

2. Literature Survey

B. F. Manly reveals that the purpose of the eBook is to introduce multivariate statistical strategies to no mathematicians. It is assumed that readers have a working information of standard statistics, inclusive of checks of significance the usage of normal, t, Chi-squared and F distributions, evaluation of variance and linear regression. The authors made an brilliant effort by means of providing multivariate records of different kinds, which include body measurements, made on or more types of people within each group and raising questions which include how exceptional the measurements are within groups and how distinctive they are between distinctive kinds of individuals. With one measurement, differences between groups is examined by means of evaluating individual mean values and variances within companies. With p measurements, p mean values are needed, and p (p − 1) variances and covariance for comparison. Appropriate multivariate

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4790-4795
https://publishoa.com
ISSN: 1309-3452

techniques for this motive have been demonstrated. In addition, there is the problem of grouping given populations by using similarity of measurements which needs a measure of distance among populations primarily based on observed facts. The authors give a very good account of various techniques to be had for these purposes. Some of the measures of similarity such as Penrose and Mahalanobis distances are noted for possible use. Penrose distance does no longer take into account correlations between measurements and might not be suitable in all practical applications. Mahalanobis distance will be suitable for correlated variables while the measurements are nearly typically distributed.

According to C. Tantithamthavorn, the prejudice and variability of version validation procedures in the domain of fault prediction are investigated in this research. According to an analysis of a hundredth and only one public defect datasets, 77% of them are extremely prone to delivering dangerous results– deciding on the right versions verification approach.

According to S. Jiang, researchers in the field of software defect prediction have been particularly interested in class imbalance. The class imbalance can affect the performance of fault prediction models in practise. The investigation will be conducted to assess the functional stability of six widely used software defect prediction models.

Contrast pattern-based classifiers, according to O. Loyola-Gonzalez, are an essential family of both intelligible and accurate classifiers. Nonetheless, those classifiers do not perform well in situations where there is a class imbalance. We present a new contrasting template classifiers for imbalanced class issues in this paper. Our solution to the problem of class imbalance combines pattern support with the imbalanced class levels at the classification phase of the classifier. There is a large disparity between both the probability of different classes in many supervised training applications, i.e., the probability with where an instance matches to the various classes of the classification. In required to practice the classifier and then classify unknown data, classification model requires previously categorised reference samples (the GT). Supervised approaches inside the area of hyper spectra image classification are classified as according their training methodology. The SVM classifier attempts to distinguish two classes using a hyper plane in which the lowest distance (referred to it as the margins) between both the training images of the two classifications is as large as possible. Support vectors are the closest spectra that are utilised to determine the hyper plane.

According to W. Lee, "we suggest a new weight correction factor that is used to a weighed SVM classifier (SVM) as a base learners of the Proposed technique to solve class imbalance in data. Different relative scores are computed and allocated to related examples by categorising involved in new on the SVM margin. While learning a weighted SVM, the adjustment factor is multiplied by the example value in the AdaBoost algorithm. Because higher body mass slides down the boat deep in the water in on-water rowing, this does not give the complete story. The erg doesn't really penalise the stronger rower in this manner, so when it comes to time to be in the boat, a bigger row may appear to have more potential than they actually do.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4790-4795
https://publishoa.com
ISSN: 1309-3452

3. Existing System

Previous research has presented models enabling fault prediction at the code level, both inside and across projects, depending on the software development. Despite the reality that these results achieved notable accuracy, stability, fault analysis, and improved binary fault identification performance, the literature survey lacks an approach for estimating the number of software defects, and no previous studies have taken the predictors used in this study into account. They employed an integrating stability model, a programme model for the prediction, a Rayleigh model, and software safety estimation to improve prediction findings. By using metrics based on antipatterns, we were able to improve the efficiency of defect prediction.

They employed refactoring to fix bad designs and anti-patterns to find design flaws that could lead to more issues in the future. If anti-pattern data could be used to discover problems, the team of developers can use refactoring to reduce the software's defect risk. By superimposing a naïve Bayes model on a prediction model, we created a predictive model with good accuracy and predictive ability.

The data utilised for categorization has unequal proportions across different classes, fault prediction studies appear to have low predictive accuracy, whereas balanced data results in improved predictive performance. To improve accuracy of defect prediction, one approach that can be utilised to address such an imbalance problem is package-based clustering.

4. Proposed System

The concept explains how we use a systematic approach to forecast the quantity of software problems. The data pre-processing stage is the first step. It ensures that detailed information regarding the data's source is gathered throughout this phase. Data analysis follows, in which we extract many measures from the datasets. In addition, for each dataset, we determine the design complexity. The effects of design complexity were compared to the effect of the chosen predictive factors in this study. The implementation of the modelling technique to generate the results is the next step in the suggested framework. The Rayleigh curve is used to model the data.

Imbalanced classes are a well-known issue in machine learning research that can influence the outcome of a prediction study. If the datasets utilised in a predicting study are properly cleaned and pre-processed, the results will be accurate, early defect prediction in the Tri Model technique can be facilitated, letting the development team to focus on attaining better outcomes. As a result, such methods can help to improve the quality of software. Recent defect prediction research have called into question the quality of dataset utilised in defect prediction, as well as the necessity of adequately pre-processing such datasets. The detailed development of the suggested modelling technique. Our modelling technique is based on the Rayleigh curve, which indicates the quantity of faults throughout the project. This graph shows how software problems change over time during the development process. If errors are not discovered and eliminated as software development progresses, the amount of errors increases. We began by looking through the datasets to determine what values the selected variables had in past projects. After that, we used these to construct and develop our

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4790-4795
https://publishoa.com
ISSN: 1309-3452

predictive model. The defect density is calculated by dividing the number of flaws by the project's size. There is no unit of measurement for defect density. The flaws is 0 at the beginning of a project. As the project progresses from a phase to the next, the likelihood of introducing defects grows. The defect rate rises as the number of phase transitions in software development grows.

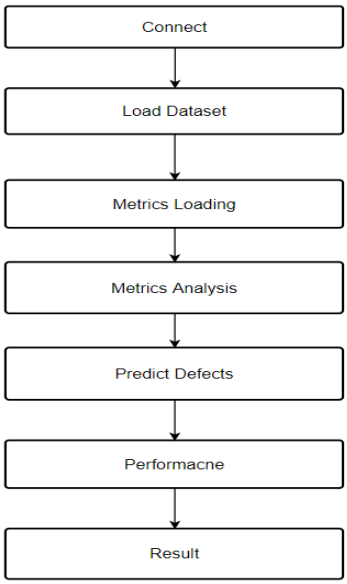

Figure1. Flow diagram of proposed work

5. Modules of Work

- Data pre-processing
- Proposed modeling technique
- Chart of the defect prediction process
- Defect prediction procedure
- Model comparison and report

A. DATA PRE-PROCESSING

The data pre processing step of our proposed architecture, that states class Imbalance and data cleanness, is the initial phase. If the data utilised in a predictive research are cleaned properly and preprocessed, early defect prediction in the Tri Model technique can be facilitated, enabling the development team to focus on attaining better outcomes. As a result, such methods can help to improve the quality of software. Recent defect prediction research have called into question the quality of dataset utilised in defect prediction, as well as the necessity of adequately preprocessing such datasets. In the experimental part, we go over our dataset preprocessing stage in more detail.

B. PROPOSED MODELING TECHNIQUE

This section explains the evolution of the suggested modelling technique. The Rayleigh curve depicts the number of defects as a function of time during a project, is the basis for our modelling technique. This graph shows how software problems change over time during the development process. If errors are not discovered and eliminated as software development progresses, the amount of errors increases. Then we modelled these variables and used them to build our model.

C. CHART OF THE DEFECT PREDICTION PROCESS

This section shows how to anticipate the amount of software defects using a basic step-by-step flow diagram. Our process is depicted in a flowchart. The diagram starts with data gathering and finishes with model evaluation to evaluate the prediction models outcomes. The data collecting stage is the first stage in the diagram, and it involves determining the data's source and format. Then there's data analysis, which includes data cleansing and any methods used to solve class imbalance, which is still a problem in machine learning and data mining research. The mean defect density for every dataset is then calculated.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4790-4795
https://publishoa.com
ISSN: 1309-3452

We double-check that all variables from data, including module, are properly evaluated before we use our modelling technique. In this work, we evaluated the datasets after they were gathered to specify the amount of defective module and then estimated the mean defect density of every dataset, following the first step of our suggested methodology.

D. DEFECT PREDICTION PROCEDURE

The procedures for building the model used to predict the amount of faults, as well as the steps for pre-processing the datasets, are presented in this section. The suggested technique can help overcome the constraints of datasets pre-processing, which is significant since prediction of software models are heavily reliant on the quality of data they are based on. First, as indicated in Equations, the predictor variables generated by our modelling technique are the input variables. The average defect density g, average defect velocity v, average defect entry time t are some of the predictor factors, are used to build our prediction models because we believe they are connected to the number of errors as a function of time.

E. MODEL COMPARISON AND REPORT

We used a cross-validation sampling method throughout our tests. Importantly, any prediction system's performance is determined by the data sampling used. Model performance estimates are used to predict how well a model performs on unknown data. For their performance evaluation, the authors used acrossvalidation sampling technique. Crossvalidation uses multiple train and test data to avoid one depending on another, and the method is known to be essentially unbiased. On the other hand, it may have more volatility. As a result, the authors of contend that when working with tiny samples, either bootstrapping or cross validation are reliable. Unstable crossvalidation outcomes, on the other hand, can be balanced through repeating the validation procedure.

6. Experimental Results

Software defect prediction gives software teams actionable results while also helping to industrial success. We've included four pieces of software in this package. We used user feedback as a source of information(dataset). Rayleigh distribution is used for predicting defects and performance. The fault and performance of one of the software are depicted in the diagram below. Similarly, the same can be done for the remaining software.

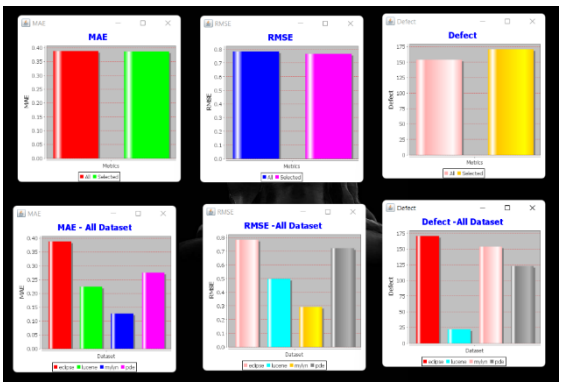

Figure2. Output

7. Conclusion

Several issues that occur in the prediction of software defects are to be resolved. As a result, we've provided a Tri Model strategy for using predictor variables to anticipate the errors. Only academic research is allowed to use translations and content mining. Personal usage is allowed, but republication necessitates defect acceleration, and we discovered a relationship between every

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 3, 2022, p. 4790-4795
https://publishoa.com
ISSN: 1309-3452

variable and the number of faults. The mean defect introduction time has a negative correlation. Managers and development teams will benefit from the proposed method's results.

As a result, project managers can concentrate on the pace at which a program progresses through one phase to another through time in order to minimise errors. The outputs of our method must be confirmed in order to verify the usefulness of our technology for defect prediction. Future study can validate this technique for forecasting the number of flaws in a new software release utilising the most up-todate datasets from any software company, as well as other predictor variables.

References

[1]. Z.-W. Zhang, X.-Y.Jing, and T.-J. Wang, "Label propagation based semisupervised learning for software defect prediction," Automated Software Engineering, vol. 24, no. 1, pp. 47–69, 2017

[2]. B. F. Manly and J. A. N. Alberto, Multivariate statistical methods: a primer. CRC Press, 2016.

[3]. C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," IEEE Transactions on Software Engineering, vol. 43, no. 1, pp. 1–18, 2017

[4]. Q. Yu, S. Jiang, and Y. Zhang, "The performance stability of defect prediction models with class imbalance: An empirical study," IEICE TRANSACTIONS on Information and Systems, vol. 100, no. 2, pp. 265– 272, 2017.

[5]. O. Loyola-Gonzalez, M. A. Medina-P ´ erez, J. F. Mart ´ ´ınez-Trinidad, J. A. Carrasco-Ochoa, R. Monroy, and M. Garc´ıa-Borroto, "Pbc4cip: A new contrast pattern-based classifier for class imbalance problems," Knowledge-Based Systems, vol. 115, pp. 100–109, 2017.

[6]. W. Lee, C.-H. Jun, and J.-S. Lee, "Instance categorization by support vector machines to adjust weights in adaboost for imbalanced data classification," Information Sciences, vol. 381, pp. 92–103, 2017

[7]. N. Ofek, L. Rokach, R. Stern, and A. Shabtai, "Fast-cbus: A fast clustering-based undersampling method for addressing the class imbalance problem," Neurocomputing, 2017.

[8]. W. Mao, J. Wang, L. He, and Y. Tian, "Online sequential prediction of imbalance data with two-stage hybrid strategy byextreme learning machine," Neurocomputing, 2017.

[9]. J. Petric, "Using different characteristics of machine learners to identify ´ different defect families," in Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering. ACM, 2016