International Journal of Research
(IJR)

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

A Practical Approach of Software Fault Prediction Using Error Probabilities and Machine Learning Approaches

Mr. Raja Lodhi

Research Scholar

Department of Computer Science & Engg.

Lakshmi Narain College of Technology, Bhopal

Prof. Rajkumar Sharma

Assistant Professor

Department of Computer Science & Engg.

Lakshmi Narain College of Technology, Bhopal

1. ABSTRACT

Software fault prediction is used to improve the testing efficiency and software quality by earlier identification of software faults associated with software. The identification of faults is usually carried out using the task of classification. The task of classification utilises the code attributes and other features to predict the fault instances. The detection of software faults is prominently affected by a poor classification decision and hence an improved decision-making model is required to predict the patterns using the attributes collected out from the datasets. In the first part of the research, the study proposes a Bayes Decision classifier associated with the finding of error probabilities and integrals in software fault prediction. This chapter discusses the fundamental software error prediction using feature and classifier data. It also discusses the proposed software error prediction with fault predictable region that includes Chernoff Bound and Bhattacharyya Bound. The proposed Bayesian decision algorithm with error probabilities and integrals of fault predictions learning model is used to predict the software faults. It works on two different bounds namely Chernoff Bound and Bhattacharyya Bound.The performance of the proposed methods is tested against several other machine learning classifier over collected software fault datasets.

Keywords: Software defect prediction, Fault Detection, machine leaning.

2. INTRODUCTION

Much of the software development budget is spent on quality control and software testing (Arar & Ayan 2015). This thus shows the value of testing during the life cycle of software development. Over the years, software systems have expanded and become more complex, making it more difficult to provide high - quality software. The aim is to provide the end user with a bug - free software. The software must be thoroughly tested and therefore an expensive, tedious and at times impossible task can be achieved in order to acquire such confidence. Resources and time constraints often limits testing. The prediction of fault - prone code enables practitioners to target the fault - prone modules with their resources and efforts, thus improving the quality of software and reducing maintenance costs and efforts (Dhankhar *et al.* 2015). Prediction of software failure permits the detect of defective code during developing software and prevents the spread of defective code in other areas of the software. It is also a process which helps optimize tests, focusing on modules which are susceptible to defects, identifying candidates for re-factoring and enhancing software quality overall. Software fault prediction can be used by project managers. During the development stage, they can measure the quality of the work by continuously measuring the module fault. By predicting failures, the project manager can detect and assign tasks based on the data and thereby improve process efficiency.

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

2.1 SOFTWARE FAULT INFORMATION

Information on software faults includes all the faults reported during the software life cycle. To store the source code version control systems and to store the reported fault data, Change Management Systems are tend to be used (Catal 2011). The datasets containing the source code are listed as available (Radjenovic *et al.* 2013). Unbalanced data is one of the main problems concerning defect data in the source dataset. Most modules are labeled non-fault susceptible while the rest of them are labeled faulty.The distributed data can therefore affect the performance of fault prediction methods. However, it is frequently recommended that the minority be examined to balance data in order to deal with this issue (Shatnawi 2012). Nevertheless, the present study uses improved SVM tool to improve the imbalanced data and on other hand in existing studies, the concern is mainly on a tool, which is intended to be used in real life without insight into the data balance.

2.2 SOFTWARE FAULT PREDICTION

A software defect or bug is defined as a state of the software that does not meet the predictions of users (Erturk & Sezer 2015). To identify the defects in the software packages, different types of explorational research are used. Testing plays a significant role in software development through the mining process in which the results are referred to as bugs (Xuan *et al.* 2015). Details of bugs are stored in a bug repository and this plays an important role in the monitoring of code errors (Geng 2018).

2.3 ROLE OF SOFTWARE FAULT PREDICTION

The Specification of Software Requirements and design documents are analyzed and developed by a user-defined code team. The coding guidelines are the focus of this team to make sure the developer codes are coherent within the project. The requirement can be divided into several modules and the modules can be developed by each team. Based on the size and requirement of the project, the number of modules and module is determined.

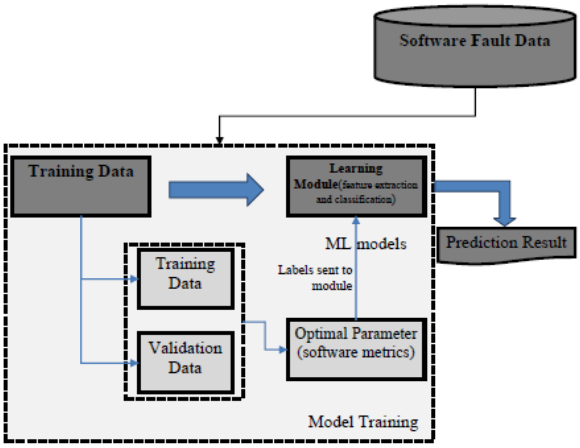


Figure 1 Software fault prediction process

The software fault data and codes are loaded in the database, which are sent as training data to the learning module. The training data also has both training and test dataset. The optimal parameter suggested provides the feedback to the learning module based on the training data. The feedback is repeated over each iteration and provides improved accuracy to the learning model. Finally, the learning module operated under machine learning operates on the test data and provides the

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

prediction results, which are evaluated under different metrics to check the efficacy of the system.

3. SOFTWARE DEFECT PREDICTION ALGORITHMS

In this section of the study the algorithms used in the prediction model of general software defects are listed with the general idea behind the prediction process of each algorithm.

- **Logistic Regression (LR)** is considered as a statistical method used to classify the dataset where the outcome is determined by one or more independent variables. One of two possible outcomes is the classification result.

- **Naïve Bayes (NB)** is regarded as a classification method based upon the Bayes rule, that determines whether an instance is subject to a certain labeling value. As final classification, the label with the highest probability is selected.

- **Random Forest (RF)**(Hong2012) is considered the method of classification consisting of a tree predictor collection each of which is used to classify an unknown event. The final classification for the unknown instance is selected based on the predictions of the trees.

- **K-Nearest Neighbour (KNN)** is considered as a decision procedure that follows a non-parametric pattern and it helps to classify an unknown instance based on its nearest neighbour.

- **Support Vector Machine (SVM)** is the classification method of machine learning. In view of a set of data labeled in which two label classes are possible, the algorithm constructs a model maping the data in a space in order to divide as wide a clear divide between the two separate classes of labeling data. Through this model the unknown data is mapped into the previously specified space and the etiquette class of the unknown data is predicted based on which side of the gap.

- **Artificial Neural Network (ANN)**or Deep Neural network (Geng 2018) is a model - based machine learning method for classifying. A model of ANN consists of layers of units known as neurons. The input - level, hidden layer and output - levels are usually called the layers. Between the input and the output layers, there may be more than a hidden layer. The ANN model learns to predict the values of unknown information by using a set of data with known labels.

3.1 EVALUATION MEASURES FOR DEFECT PREDICTION MODELS

Some of the measurements used to evaluate the performance of the prediction of software defects covered by Chapter 5 will be listed in this section of work and described. For model predictions of software defects, after predictions have been made on whether the entity is deficient or clean, four possible results exist for an entity. The prediction results parameters are the shown as follows:

- True Positive: A defective entity is classified as defective

- False Negative: a defective entity is classified as clean

- True Negative: a clean entity is classified as clean

- False Positive: a clean entity is classified as defective

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

Based on these results, actions to evaluate the precision of the prediction model of a software defect are defined. In the study covering the software defect prevention model, the most common measure used to evaluate the performance of a defect prediction model is the precision and recall and F-measure.

4. LITERATURE REVIEW

Dejaeger *et al.* (2021) adds to the coding by thinking about 15 diverse Bayesian Network (BN) classifiers and contrasting them with other mainstream machine learning strategies. Besides, the relevance of the Markov cover guideline for highlight choice, which is a characteristic expansion to BN hypothesis, is researched. The outcomes, both as far as the AUC and the as of late presented H-measure, are thoroughly tried utilizing the factual structure of Demšar. It is inferred that straightforward and fathomable systems with less hubs can be built utilizing BN classifiers other than the Naive Bayes classifier. Besides, it is discovered that the parts of understandability and prescient execution should be offset, and furthermore the advancement setting is a thing which ought to be considered amid model choice.
Abaei *et al.* (2021) proposed a robotized codingfault discovery show utilizing semi-regulated cross breed self-sorting out guide.

Rathore & Kumar (2021) assessed and looked at a plenty of fault prediction methods by differing the setting as far as space data, qualities of data, multifaceted nature, and so on. Be that as it may, the absence of an acknowledged benchmark makes it hard to choose a fault prediction strategy for a specific setting of prediction. The study models a suggestion framework that encourages the choice of fitting technique(s) to fabricate fault prediction model.

Bennin *et al.* (2022) presented MAHAKIL, a novel and proficient engineered oversampling approach for codinginaccurate datasets that depends on the chromosomal hypothesis of legacy. Misusing this hypothesis, MAHAKIL deciphers two particular sub-classes as guardians and produces another example that acquires distinctive characteristics from each parent and adds to the assorted variety inside the data appropriation.

Wang & Zhang (2022) used a deep learning model dependent on the repetitive NN (RNN) encoder–decoder to foresee the quantity of faults in coding and survey coding dependability. The deep learning NN demonstrate develops the layer levels as well as adjust to catch the preparation attributes. A far reaching, top to bottom experiment and highlight uncovering eventually demonstrates the model can have reasonable prediction performance. Experimental results demonstrate that the proposed model has better prediction execution contrasted and other parameter and NN models.

An *et al.* (2020) dissected the execution of nine broadly utilized machine learning classifiers—Bayes Net, NB, artificial neural system, SVM, KNN, AdaBoost, Bagging and RF for coding deficiency prediction. Two standard testing systems—SMOTE and Resample with substitution are utilized to deal with the class lopsidedness fault. The study further utilized FLDA-based component determination approach in mix with SMOTE and Resample to choose most discriminative measurements. At that point the best the classifiers dependent on execution are utilized for coding fault prediction. The experimentation is done more than 15 publically accessible datasets (little, medium and vast) which are gathered from PROMISE store. The proposed Resample-FLDA technique gives better execution when contrasted with existing strategies as far as accuracy, review, f-measure and zone under the curve.

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

Kumar *et al.* (2019) exhibited in his study includes building a compelling errorprediction instrument by recognizing and examining the prescient intensity of a few understood and broadly utilized coding measurements for fault prediction. The study apply ten distinctive element choice strategies to pick the best classification of measurements from a lot of twenty source code measurements. The study construct the fault prediction demonstrate utilizing Least Squares SVM learning technique related with direct, polynomial and outspread premise work bit capacities. The study perform investigates 30 Open Source Java ventures. Exploratory outcomes uncovers that the prediction model is best reasonable for tasks with faulted classes not exactly the edge esteem contingent upon fault ID proficiency.

Miholca *et al.* (2018) built up a novel supervised clustering strategy called HyGRAR for coding in accurate prediction. HyGRAR is a rule mining based ANN to separate among inadequate and non-errorcodes. Experiments performed dependent on 10 open-source informational collections showed the phenomenal execution of the HYGRAR classifier. HyGRAR performed superior to anything a large portion of the recently proposed methodologies for coding defect prediction in execution evaluations utilizing similar informational indexes.

Arshad *et al.* (2018) proposed a semi-superviseddeep fuzzy C- means (DFCM) clustering for coding error prediction, which is the cumulation of semi-administered DFCM bunching and include pressure methods. Deep is used for the component based multi bunches of unlabeled and named informational indexes alongside their named classes. In the methodology, for the preparation show, The study at the same time manage the unsupervised data and regulated data to abuse the obnubilated data from unlabeled data to marked data to help the development of the exact model. The study use DFCM bunching to deal with the class irregularity fault and withal fuzzy rule is much the same as human rule. The study further enhance the prediction execution with the combination of highlight learning methods include extraction and highlight choice to produce great highlights to diminish the larger data for classification. Nonetheless, by the execution of the model outcomes, the amalgamation of deep multi clusters and highlight procedures work better because of their capacity to distinguish and amalgamation basic data in data include. The order demonstrate is predicted on the most extreme homogeneous between the highlights of named and unlabeled data, the model is prepared on the un-boisterous informational collection gotten by the deep combination of multi clusters and highlight strategies. To check the adequacy of the methodology, the study picked informational collections from genuine coding venture (NASA and Eclipse), and after that the study contrasted the methodology and various most recent traditional gauge techniques, and research the execution by utilizing execution estimates, for example, likelihood of discovery, F-measure, and area under the curve.

Manjula & Florence (2018) presented a cross breed approach by joining hereditary algorithm (GA) for highlight optimization with Deep Neural Network (DNN) for classification. An improved adaptation of GA is joined which incorporates another system for chromosome structuring and wellness work algorithm. DNN system is additionally ad libbed utilizing versatile auto-encoder which gives better portrayal of chose coding highlights. The improved softwareivity of the proposed half breed approach because of organization of enhancement procedure is shown through contextual analyses. A test ponder is done for coding defect prediction by considering PROMISE dataset utilizing MATLAB apparatus. In this experiment, the study have utilized the proposed novel technique for clustering and inaccurate prediction.

Arora & Saha (2018) proposed a mixture SFP model manufactured utilizing Firefly Algorithm (FA) and Artificial Neural Network (ANN), alongside an observational experiment with GA and PSO based transformative strategies in advancing the association loads of ANN. Seven diverse datasets

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

were included and MSE and the perplexity network parameters were utilized for execution evaluation. The outcomes have demonstrated that FA-ANN model has performed superior to the hereditary and molecule swarm improved ANN fault prediction.

Periasamy & Mishbahulhuda (2017) developes a defect prediction (clustering, classification and association rule) model which reduces defects and helps remove the errors that terminate with a quality software system. The correct prediction of software software bugs contributes to software quality during software testing and facilitates maintenance through clustering, grading and association rules.

HaraldAltinger *et al*. (2017) describes a software development error and has determined that the developed model suffers from a robust unequaled distribution to a low bug rate. Due to low predictive performance, the turning parameters were considered necessary. The study has made the Fault prediction on a larger dataset with an imbalanced class distribution range of 2.63 to 14.89%.

Table 1 Methods used for Software Fault prediction

Author	Methods
Moeyersoms *et al.* (2021)	RF and SVM
Bennin *et al.* (2022)	presented MAHAKIL
Wang & Zhang (2021)	RNN
Kalsoom A et al. (2018)	Bayes Net, NB, ANN, SVM, KNN, AdaBoost, Bagging, and RF
Kumar *et al.* (2019)	Least Squares SVM
Miholca *et al.* (2018)	HyGRAR
Arshad *et al.* (2018)	semi-supervised deep fuzzy C-means
Manjula & Florence (2018)	Deep neural network with GA
Arora & Saha (2018)	ANN

5. SOFTWARE ERROR PREDICTION

Prediction of software errors measures the process of software development to create predictions. The aim of the prediction of software errors is to predict how likely a certain portion of a project tends to fail. For code lines or a development packages or a component assembly, the probability can be estimated. For such purposes, this information may be used:

- Targeting the reviews in code,
- Targeting the testing patterns and
- Targeting the communication in each project

As an example, a N-variable linear regression error prevention is displayed in Equation (3.10). The probability of errors is Y, and the values of the characteristics are X. The most likely cause of error would be the value of the estimated weights β.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_N X_N + \square_i \quad (3.1)$$

Implementation error prediction can be seen as a controlled classification problem for machine

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

learning. Controlled machine learning is a practice based on historical data of training a machine learner. The trained student will then be able to classify the actual data.

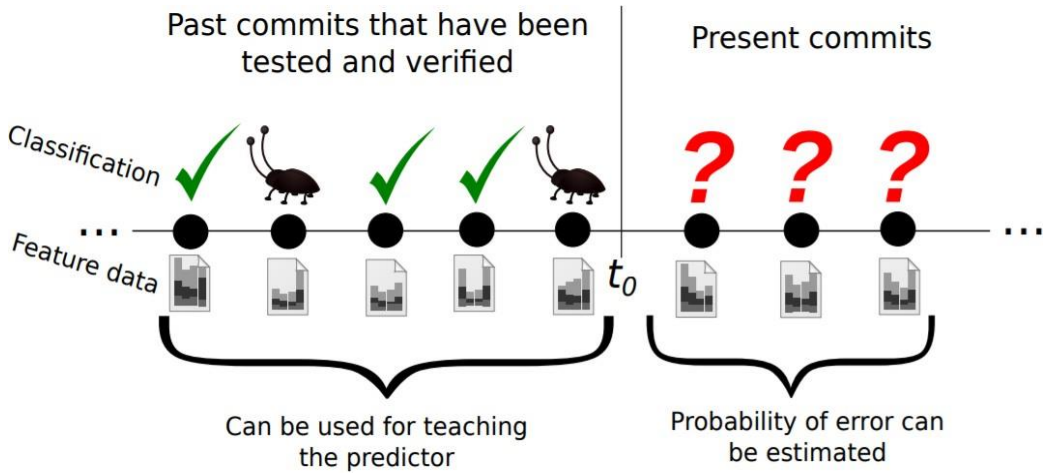


Figure 2: Prediction of faultiness using machine learning in Software systems

A commit is the basic unit of the history of software development. The features are removed and the classification is done for the commits. Figure 3.1 shows the process of error prediction.

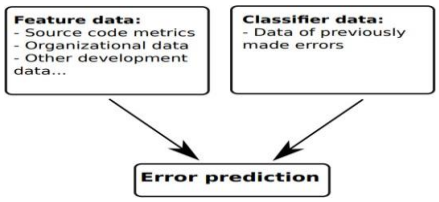


Figure 3 Machine learning for error prediction

Training data typically represent the history of the project in the prediction of software errors. There was also an investigation into the use of combined history from earlier projects (Zimmermann *et al.* 2009; Turhan *et al.* 2009). Feature and classification data required for error prediction. In a typical software project, information on the code, organization and history of development is features which can be extracted. Classification data are data that were deficient in parts of the project. An error predictor can be created using both the function and the classifier data. The Figure 3 gives this idea.

5.1 PROPOSED SOFTWARE ERROR PREDICTION

To describe the conditional probability of an event, Beyers theorem is based on the prior knowledge of possible conditions associated with the event. Based on conditional priority, conditional probability removes the unwanted pertinent event, since the Beyers rule covers "less error".

5.2 ALGORITHMS FOR ERROR PROBABILITIES

Bayesian decision theory (Bouguila*et al.* 2008) is a basic statistical approach in relation to the pattern classification problem. This approach is based on the algorithm of trade - offs with

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

probability between different classification decisions and their costs. It assumes that the problem of decision is probabilistically posed and all of the corresponding probability values are known.We are developing in this chapter the foundations and how the theory can be considered simply to formalize common meanings processes and account the problems which arise when the structure of probabilism is not fully understood.

5.3 DECISION RULE

Where such little information is required to make a decision, the following rule of decision seems logical.

Decide ω_1 if $P(\omega_1) > P(\omega_2)$; otherwise decide ω_2. (4.1)

If we have to judge a failure, this rule makes sense, but to judge many failures, it seems strange to use this principle repeatedly. After all, the study always decide the same although it is known that there are both types of decisions i.e. faults or healthy.This works well based on previous probability values. If the value of $P(\omega_1)$ is greater than the value of $P(\omega_2)$, $\omega_1$is selected for entireoperation. If the value of $P(\omega_1)$ is similar to the value of $P(\omega_2)$, the selection of probability will have fifty-fifty chance to acquire fruitful results. If the value of $P(\omega_1)$ is lesser than the value of $P(\omega_2)$ i.e. the error probability is the smaller in $P(\omega_1)$than $P(\omega_2)$,no optimal decisions can be made and the correctness of decision cannot be yield at this condition.

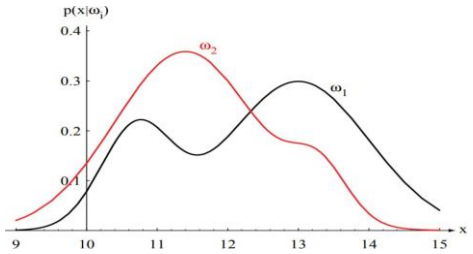
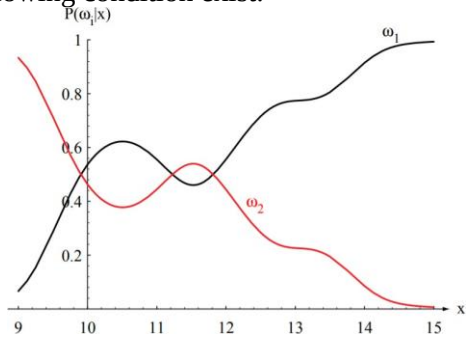


Figure 4. Hypothetical conditional class PDF that shows the probability density

Note that the product prior probability and likelihood is considered as the important factors for finding the posterior probability, where the evidence factor is given as$p(x)$ and this referred as a scale factor, which helps in guaranteeing that the sum of all the posterior probabilities is unity, where all the probabilities should be of a good one. On other hand, conditional class PDF has the condition $P(\omega_1|x) < P(\omega_2|x)$, this offers the true state of nature to be ω_2. These claims are justified by the suitable estimation of error probability before a decision is made. Hence, if a specific variable x is observed, the following condition exist.



Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

Figure 5. Posterior probabilities of P(ω1) and P(ω2)

The error probability of a variable x can be reduced by suitably deciding the value of the state of nature ω_1, upon a condition $P(\omega_1|x) > P(\omega_2|x)$ and ω2 for the condition $P(\omega_1|x) < P(\omega_2|x)$, otherwise. It is seen from the graph that the values do not occur twice for a variable *x*. This reduces the average error probability and it is expressed as follows:

5.4 PERFORMANCE EVALUATION

This section discusses the experimental setup needed to evaluate the proposed system. The experiment is performed on a 3.6GHZ quad-core processor with windows platform. A 10-fold cross-validation is used to carry out the performance evaluation on software failure dataset. The performance of proposed method is compared against conventional ensemble methods with weighted SVM classifier and random forest classifier. The proposed feature selection method using genetic algorithm is used to test the hypothesis with and without PCC. The proposed classifier model is evaluated against software fault datasets that are available publicly, which includes Camel-1.6, Ant-1.7, MC1, KC3 datasets, PC4 and PC2. The information of the collected datasets is given in Table 1.

Table 1 Information from the Defect Dataset

Dataset	Number of Components	Total number of Defective components
Camel-1.6	965	188
Ant-1.7	745	166
MC1	1988	46
KC3	200	36
PC4	1287	177
PC2	1585	16

Table 2 Software Metrics considered for evaluation in the present study

Code	Metric
M_1	Decision count
M_2	Cyclomatic complexity
M_3	Global data density
M_4	Halstead difficulty
M_5	Halstead content
M_6	Maintenance severity
M_7	Coupling between objects
M_8	Number of unique operands
M_9	Afferent couplings
M_10	Essential density
M_11	Global data complexity
M_12	Efferent couplings

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

M_13	Lack of cohesion of methods
M_14	Condition count

RESULTS AND DISCUSSION

In this section, we present the analysis of proposed method and its related discussions. The proposed ensemble learning method has multiple SVM classifier, where the final classification result is an average value of output probabilities from all individual SVM classifiers. The proposed ensemble model with software failure dataset is tested against existing classifiers like random forests classifiers and Weighted SVM that includes the preprocessing and feature selection with or without GA. The Figure 4 presents the AUC measures of these three classifiers. The result shows that proposed Bayes classifier model achieves higher AUC measure than Decision Tree and KNN. From the results, it is interesting to see that proposed Bayes classification method outperforms Decision Tree and KNN even in the presence of imbalanced dataset. The proposed method has improved results due to the use of decision rule with Bayesian Classifier with Chernoff Bound and Bhattacharyya Bound this has enabled the proposed model to alter its decision for acquire improved class samples. It could be inferred that averaging of classification outcomes compensates the presence of errors. Additionally, the proposed model behaves with better robustness against the imbalanced datasets. The robustness is further improved even in the presence of irrelevant and redundant features and this recommends the use of averaging ensembles models for software failure classification.

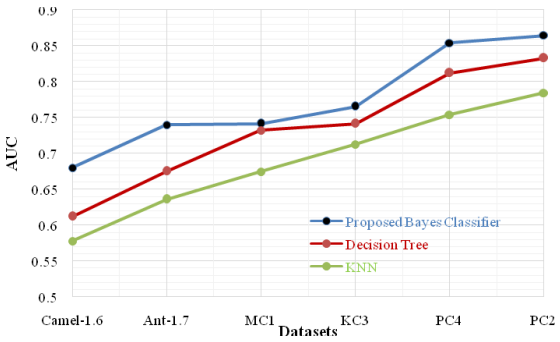


Figure 5 AUC measures of these three classifiers for Software Fault datasets

Further, it is interesting to note that use of Chernoff Bound and Bhattacharyya Bound provides improved result in proposed method than other methods for all datasets. The use of Bhattacharyya Bound has further improvised the AUC obtained under chernoff bound. From this discussion it is clear that clustering of classifier result improves the performance of overall classification result, even if other classifiers performs poor. The Figure 4.4 shows the classification accuracy between the proposed Bayes classifier and existing Decision Tree and KNN classifer.It is claimed from the results that in terms of classifiers accuracy, the proposed model outperforms other single classifiers namely Decision Tree and KNN classifer. This is true when the selection of base classifier using Decision Tree is made based its optimal performance. It is further noted that overall performance of proposed classification model is not affected by the base classifiers than other classifiers.

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

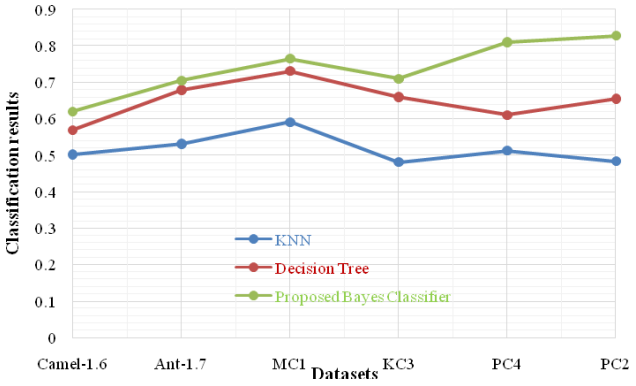


Figure 6 Classification results of these three classifiers for Software Fault datasets

This states that various algorithms and methods have examined and forecast the software fault prediction. The focus of this research is on a detailed analyze of fault prediction using error probability methodology and integral method for predicting the error that occurred during coding stages. Developers at certain times fail to detect the error by coding. Hence, in the proposed method the mathematical derivations are used with boundary conditons to identify the fault. The Chernoff Bound and Bhattacharyya Bound methods provides its improved support to predict the defects or faults. These defects are used to cluster the failures and defects, which are saved for precautionary predictions in the software repository. This is used when the testing team uses software that develops the life cycle.

SCOPE OF THE FURTHER RESEARCH

Further research is necessary to adopt the agile-based software fault prediction. In future studies, several robust software metrics must be used for the development of failure prediction models to detect the differences between the two software versions. Some studies of fault forecasting have shown that a few modules contain most of the software projects faults. In future, such studies should be inteneded to increase the information on fault prediction models. Researchers in future should carry out more studies using machine learning or heuristic or meta-heuristic or evolutionary algorithm to assess the best way to make optimal use of such approaches. Such research can significantly influence the performance of the fault prediction. Future studies can attempt to develop fault prediction models for the purpose of cross-project prediction that is considered to be useful to an organization with insufficient project history of fault diagnosis.

CONCLUSIONS

The study worked on various issues associated with the software fault prediction activities like fault prediction, software metrics, issues in data quality and performance evaluation measures. The study highlights different methodological issues and challenges linked to these software fault prediction activities. The high dimensionality of data and data class imbalance associated with software quality issues is investigated widely using feature extraction and classification, respectively. The performance is compared in terms of different metrics to evaluate the accuracy of prediction performance. The study also identified challenges for researchers to explore in the future to further develop the software fault prediction process.

REFERENCES

1. Abaei, G, Selamat, A & Fujita, H 2015. ‘An empirical study based on semi-supervised hybrid self-

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

organizing map for software fault prediction. Knowledge-Based Systems', vol. 74, vol. 28-39.

2. Alonso, J, Torres, J, Berral, JL & Gavalda, R 2010. 'Adaptive on-line software aging prediction based on machine learning', In 2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN), pp. 507-516.
3. Altinger, H, Herbold, S, Schneemann, F, Grabowski, J & Wotawa, F 2017, 'Performance tuning for automotive software fault prediction. In 2017 IEEE 24th International Conference on Software Analysis', Evolution and Reengineering (SANER), pp. 526-530.
4. Andersson, Carina 2007, 'A replicated empirical study of a selection method for software reliability growth models', Empirical Software Engineering, vol. 12, no. 2, pp. 161.
5. Arar, ÖF & Ayan, K 2015. 'Software defect prediction using cost- sensitive neural network,' Applied Soft Computing, vol. 33, pp. 263-277.
6. Arora, I & Saha, A 2018, 'Software fault prediction using firefly algorithm. International Journal of Intelligent Engineering Informatics, vol. 6, no. 3-4, pp. 356-377.
7. Arora, I, Tetarwal, V & Saha, A 2015. 'Open issues in software defect prediction', Procedia Computer Science, vol. 46, pp. 906-912.
8. Arshad, A, Riaz, S, Jiao, L & Murthy, A 2018. 'Semi-supervised deep fuzzy c-mean clustering for software fault prediction. IEEE Access', vol. 6, pp. 25675-25685.
9. Bachmann, Adrian, Christian Bird, Foyzur Rahman, PremkumarDevanbu & Abraham Bernstein 2010', 'The missing links: bugs and bug-fix commits', In Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering, pp. 97-106. ACM.
10. Bai, CG, Hu, QP, Xie, M & Ng, SH 2005. 'Software failure prediction based on a Markov Bayesian network model', Journal of Systems and Software, vol. 74, no. 3, pp. 275-282.
11. Bennin, KE, Keung, J, Phannachitta, P, Monden, A & Mensah, S 2018, Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. IEEE Transactions on Software Engineering, vol. 44, no. 6, pp. 534-550.
12. Bibi, S, Tsoumakas, G, Stamelos, I & Vlahavas, IP 2006, 'Software Defect Prediction Using Regression via Classification', In AICCSA, pp. 330-336.
13. Bishnu, PS & Bhattacherjee, V 2012, 'Software fault prediction using quad tree-based k-means clustering algorithm. IEEE Transactions on knowledge and data engineering', vol. 24, no. 6, pp. 1146-1150.
14. Bouguila, Nizar, Jian Han Wang & Ben Hamza, A 2008, 'A bayesian approach for software quality prediction', In 2008 4th International IEEE Conference Intelligent Systems, vol. 2, pp. 11-49.
15. Can, H, Jianchun, X, Ruide, Z, Juelong, L, Qiliang, Y & Liqiang, X, 2013. 'A new model for software defect prediction using particle swarm optimization and support vector machine', In 2013 25th Chinese Control and Decision Conference (CCDC), pp. 4106-4110.
16. Cao, Q, Sun, Q, Cao, Q & Tan, H 2015, 'Software defect prediction via transfer learning based neural network', In 2015 First International Conference on Reliability Systems Engineering (ICRSE), pp. 1-10.
17. Catal, C & Diri, B 2009, 'Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. Information Sciences', vol. 179, no. 8, pp. 1040-1058.
18. Catal, C 2011, 'Software fault prediction: A literature review and current trends', Expert systems with applications, vol. 38, no. 4, pp. 4626-4636.
19. Catal, C, Sevim, U & Diri, B 2011, 'Practical development of an Eclipse-based software fault prediction tool using Naive Bayes algorithm. Expert Systems with Applications', vol. 38, no. 3, pp. 2347-2353.
20. Catal, Cagatay & Banu Diri 2009, 'A systematic review of software fault prediction studies',

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

'Expert systems with applications, vol. 36, no. 4, pp. 7346-7354.

21. Challagulla, VUB & Bastani, FB, Yen, IL & Paul, RA 2008. 'Empirical evaluation of machine learning based software defect prediction techniques', International Journal on Artificial Intelligence Tools, vol. 17, no. 02, pp. 389-400.
22. Choudhary, Garvit Rajesh, Sandeep Kumar, Kuldeep Kumar, Alok Mishra & Cagatay Catal 2018, 'Empirical analysis of change metrics for software fault prediction', Computers & Electrical Engineering, vol. 67, pp. 15-24.
23. Claesen, Marc, Frank De Smet, Johan AK Suykens & Bart De Moor 2014, 'Ensemble SVM A library for ensemble learning using support vector machines', The Journal of Machine Learning Research, vol. 15, no. 1, pp. 141-145.
24. De Carvalho, AB, Pozo, A & Vergilio, SR 2010, 'A symbolic fault- prediction model based on multiobjective particle swarm optimization. Journal of Systems and Software', vol. 83, no. 5, pp. 868-882.
25. Dejaeger, K, Verbraken, T & Baesens, B 2013, 'Toward comprehensible software fault prediction models using bayesian network classifiers', IEEE Transactions on Software Engineering, vol. 39, no. 2, pp. 237-257.
26. Dhankhar, S, Rastogi, H & Kakkar, M 2015, 'Software fault prediction performance in software engineering', In 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 228-232.
27. Duda, Richard O, Peter E Hart & David G, 'Stork Pattern classification. John Wiley & Sons, 2012.
28. Erturk, E & Sezer, EA 2016a. 'Iterative software fault prediction with a hybrid approach', Applied Soft Computing, vol. 49, pp. 1020-1033.
29. Erturk, E & Sezer, EA 2016b. 'Software fault prediction using Mamdani type fuzzy inference system', International Journal of Data Analysis Techniques and Strategies, vol. 8, no. 1, pp. 14-28.
30. Erturk, Ezgi & Ebru Akcapinar Sezer 2015, 'A comparison of some soft computing methods for software fault prediction', Expert systems with application, vol. 42, no. 4, pp. 1872-1879.
31. Geng, Wang, 2018, 'Cognitive Deep Neural Networks prediction method for software fault tendency module based on Bound Particle Swarm Optimization', Cognitive Systems Research, vol. 52, pp. 12-20.
32. Gondra, I 2008, 'Applying machine learning to software fault- proneness prediction. Journal of Systems and Software, vol. 81, no.2, pp. 186-195.
33. Goyal, R, Chandra, P & Singh, Y 2014, 'Suitability of KNN regression in the development of interaction based software fault prediction models. IeriProcedia, vol. 6, pp. 15-21.
34. Gyimothy, T, Ferenc, R & Siket, I 2005. 'Empirical validation of object-oriented metrics on open source software for fault prediction. IEEE Transactions on Software engineering', vol. 31, no. 10, pp. 897-910.
35. Hammouri, A, Hammad, M, Alnabhan, M & Alsarayrah, F 2018, 'Software bug prediction using machine learning approach. International Journal of Advanced Computer Science and Applications', vol. 9, no. 2, pp. 78-83.
36. Han, J, Kamber, M & Pei, J 2012, 'Data mining, concepts and techniques, Waltham, MA. Morgan Kaufman Publishers, vol. 10, pp. 978-1.
37. Hatton, L 2008, 'The role of empiricism in improving the reliability of future software. In Testing: Academic and Industrial Conference.
38. He, H & Garcia, EA 2008, 'Learning from imbalanced data. IEEE Transactions on Knowledge & Data Engineering, vol. 9, pp. 1263-1284.
39. He, Q, Shen, B & Chen, Y 2016. 'Software defect prediction using semi-supervised learning with change burst information', In 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC) vol. 1, pp. 113-122.

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Vol. 11 Issue 05
May 2024

40. Hong, E 2012, 'Software fault-proneness Prediction using random forest. International Journal of Smart Home, vol. 6, no. 4, pp. 147-152.
41. Hu, QP, Xie, M, Ng, SH & Levitin, G 2007, 'Robust recurrent neural network modeling for software fault detection and correction prediction. Reliability Engineering & System Safety, vol. 92, no. 3, pp. 332-340.
42. Jiang, Y, Lin, J, Cukic, B & Menzies, T 2009. 'Variance analysis in software fault prediction models', 20th International Symposium on Software Reliability Engineering, pp. 99-108.
43. Kalsoom, A, Maqsood, M, Ghazanfar, MA, Aadil, F & Rho, S 2018. 'A dimensionality reduction-based efficient software fault prediction using Fisher linear discriminant analysis (FLDA)', The Journal of Supercomputing, vol. 74, no. 9, pp. 4568-4602.
44. Kan, SH 2002, 'Metrics and models in software quality engineering. Addison-Wesley Longman Publishing Co., Inc..
45. Kanmani, S, Uthariaraj, VR, Sankaranarayanan, V & Thambidurai, P 2007. 'Object-oriented software fault prediction using neural networks. Information and software technology', vol. 49, no. 5, pp. 483-492.
46. Kaur, Rajdeep & ErSumit Sharma, 2018, 'Various techniques to detect and predict faults in software system: survey', International Journal on Future Revolution in Computer Science and Communication Engineering (IJFRSCE), vol. 4, no. 2, pp. 330-336.
47. Kim, S, Zimmermann, T, Whitehead Jr, EJ & Zeller, A 2007, 'Predicting faults from cached history. In Proceedings of the 29th international conference on Software Engineering (pp. 489-498). IEEE Computer Society.
48. Kumar, L, Sripada, SK, Sureka, A & Rath, SK 2018, 'Effective fault prediction model developed using least square support vector machine (lssvm). Journal of Systems and Software, vol. 137, pp. 686-712.
49. Lee, T, Nam, J, Han, D, Kim, S & In, HP 2011, 'Micro interaction metrics for defect prediction'. In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering, ACM, pp. 311-321.
50. Li, Z, Jing, XY & Zhu, X 2018, 'Heterogeneous fault prediction with cost-sensitive domain adaptation'. Software Testing, Verification and Reliability, vol. 28, no. 2, pp. e1658.
51. Lu, H, Cukic, B & Culp, M 2012, 'Software defect prediction using semi-supervised learning with dimension reduction'. In Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, ACM, pp. 314-317.
52. Malhotra, R & Jain, A 2012, 'Fault prediction using statistical and machine learning methods for improving software quality'. Journal of Information Processing Systems, vol. 8, no. 2, pp. 241-262.
53. Manjula, C & Florence, L 2018, 'Deep neural network based hybrid approach for software defect prediction using software metrics'. Cluster Computing, pp. 1-17.
54. Martarelli, NJ & Nagano, MS 2018, 'A constructive evolutionary approach for feature selection in unsupervised learning'. Swarm and Evolutionary Computation, vol. 42, pp. 125-137.
55. Menzies, T, Greenwald, J & Frank, A 2007, 'Data mining static code attributes to learn defect predictors'. IEEE transactions on software engineering, vol. 33, no. 1, pp. 2-13.
56. Miholca, DL, Czibula, G & Czibula, IG 2018, 'A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks'. Information Sciences, vol. 441, pp. 152-170.

Received: *25 April 2024*
Revised: *7 May 2024*
Accepted: *15 May 2024*

DOI: https://doi.org/10.5281/zenodo.11195244