Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/13S121

Investigation Analysis for Software Fault Prediction using Error Probabilities and Integral Methods

S. Karuppusamy[1,*] *and G. Singaravel*[2]

[1] Computer Science and Engineering, Nandha Engineering College, Erode, Tamilnadu, India
[2] Department of Information Technology, K.S.R College of Engineering (Autonomous), Tiruchengode, India

Received: 2 Nov. 2018, Revised: 1 Dec. 2018, Accepted: 5 Dec. 2018
Published online: 1 Aug. 2019

Abstract: In this paper, in-depth analysis of faults in the code phase is detected through integral methods that identify the error in software. The repositories of the data set are collected during the software product development life cycle model, which is then integrated with a machine learning algorithm namely Bayesian decision theory to detect the error probabilities and to predict unbound error during the prediction of the software faults. In prior, the faults are predicted in repository for a given data set using error probability and error integral method that identify the probability of error and correction, which is then applied with Gaussian method to find the levels of the error probability with minimum and maximum integral of acceptable faults in the repository.

Keywords: Software fault prediction, repository mining, error probability, integral method

1 Introduction

In the domain of software testing, bugs play a major role that could be termed Software Defect. This is classified into various aspects based on their performance viz., error, flaw, failure or fault. The bug occurs due to the unexpected use of the customers. The bugs are predicted through software testing, which in turn is referred as Software defect prediction that predicts the defective modules in the code developed. The majority of machine leading programs are incapable of extracting defects from the database. The supervised learning is used to identify defect and predict at the logical level, where the code sample is used in the test improving the performance of defect prediction [1,2].

Artificial models [3,4,5,6] are used recently to solve a wide range of computation applications hard to be solved using analytical and numerical computations. In this paper, we focus on the fault prediction rate in the product life cycle model by using Bayesian decision theory [7] as one of machine-learning models with the help of large historical data set [9]. Bayesian decision theory is a basic concept for statistical loom for the problem to find the pattern matching of fault predication with reference to the existing stored data in repository in order to find the feasible solution of mining the data. Classification of decision theory helps to minimize the error probability using integral of fault prediction [10]. Px(x/W1) indicates that the density task for accidental variable of X, given in data set, uses Bayes to find the source of error in software coding.

In software development life cycle, testing plays a major role in which fault identification is carried out through the mining methods, where the results are termed as bugs. A bug repository has stored details about the bugs and it plays a significant role in supervising software bugs in coding [22]. The challenges are focused on the problem raised during the code tested. The tested results are being stored in the repository, which help in handling to handle the software development tasks in the future by comparing the faults raised during the development.

A software defect and bug is a state in a software product which does not encounter the end-user expectations [11]. The different types of exploratory research are used to identify the faults from the software packages.

Ubhiand in 2017 [11] found that data mining methods have reduced the fault-prone real-time system earlier to the testing phase of the software development life cycle. The software mining methods have been evolved for the purpose of enhanced evaluation methods, fault tolerance, to rationalize the decisions for efficient quality attributes

* Corresponding author e-mail: karuppasamyitphd@gmail.com

and by enabling to take better implications from the mining approaches. Naheed Azeem in 2011 [1] suggested the unsupervised learning framework to find out the best algorithm for defect forecasting. Furthermore by analyzing the effect of feature selection and the cost, responsive learning procedures can be used to develop better defect prediction replicas. Julie in 2014 [12] discussed the software liability and effort forecast which is a significant task to curtail costs of a software project and to forecast the time for completion with fault prone model. By using various data mining techniques, the analytical models are not only precise but also comprehensible. By applying the mining method, the ALPA is used to progress the rule set in terms of fidelity. Wei Fu in 2017 [2] revisited the unsupervised model by approaching the quality data from software schemes, which are time-consuming and expensive due to collection of unlabeled data. The supervised method suggested by the author is used to identify the defective or non-defective codes that are selected as best alternate techniques for labelled data. The supervised forecasters are one way method that are suitable to select the best model for defect prediction. Kharche in 2017 [9] allocated the correct potential developer for bug fixing, which has been done through the machine-learning algorithm, termed as bug Triage. The machine-learning algorithm automates the bug triage to help the organization to minimize the time and cost, thus improving the quality of model by dealing with bugs. The best tentative outcome shows that grouping of CHI and ICF yields the best accuracy rate, which is optimal for training set reduction. This is considered as a powerful tool through machine learning algorithm for training set of bug triage to improve the software quality. Viji in 2017 [10], described the software fault prediction that plays a significant role in the software reusability which results in software quality, this reduces the time and cost for software testing. The software fault prediction predicts the fault based on historical data and different machine learning techniques form the repository. The k-mean method is used for discretization. Earlier, Apriori algorithm is used to generate rule in data mining for larger data set. The combination of apriori algorithm and K-Means method is used to minimize the faults in dataset. There was a focus on elaborating machine learning to improve the accuracy of predicting software defects in the code. Periasamy in 2017 [13] uses defect prediction model (clustering, classification and association rule) that reduces the defects and contributes to the removal of the bugs ends with a quality software system. The correct prediction of bugs in the software product during the software testing contributes to the product quality and eases the maintenance by the clustering, classification, and the association rule.

Harald Altinger in 2017 [14] describes the fault in developing software that suffers from a robust imbalanced distribution to a low bug rate in the developed model. The turning parameters are required for predictive low performance.

Awni Hammouri in 2018 [15] found that Software Bug Prediction (SBP) played a key role in the fault free detection of the software development and maintenance. The machine learning algorithm with the supervised data set has been used for the fault prediction. The fault prediction can be calculated mathematically through the Gaussian bound, Chernoff bound, and Bhattacharyya bound by proving the error prediction rate at product life cycle [17]. Furthermore for deeper analysis of the fault, machine learning models and data mining techniques [8, 19,6,25,26] are used for software repositories to extract the flaws in the software product.

Project team is not tested and handed over to customer, the customer receives the faulty product at the end. Error may start to creep in the system from the phases requirements onwards, which then falls at coding phase. Risk is a likelihood, which is a program fault and this results to a negative impact on the business. Defects are also called as Fault or Bug in IT industry and the following list provides the definition of a Defect.

- A defect is the mismatch between the program and its specification of the requirement of an error, if and only if the specification exists and is correct to the given requirement.
- For the lack of desire for completion or perfection of the coding; a deficiency occurs.
- The shortcoming prevents an item from being complete, desirable, effective, safe, or of merit, which makes it to malfunction or fail in its purpose.
- A defect is an imperfection that causes inadequacy or failure; a shortcoming to fail.

Debugging process can make the test cases execute and its test execution results are assessed. Then the differences between the actual and the expected behavior is identified and analyzed. Once the root cause of the problem is diagnosed, the developer corrects and removes the errors from the system [16,18]. Care should be taken while performing the debugging process to make sure that the error is properly removed and it does not introduce any new error in the system.

Basically the two outcomes of the debugging process are

- Error is properly diagnosed and the cause will be found for the error, which is then removed from the system
- Error is not properly diagnosed and hence the cause is not found so the error/fault is not removed from the system

Debugging process requires more concentration and sound knowledge on the entire architecture of the program which helps the debugging personnel to identify where exactly the problem lies in the system and the solution is provided to overcome such faults [20]. Few developers are very good at debugging with the analytical and logical thinking mind set and some people don't have

those skills for debugging.

Developers are not good in debugging start this activity to correct the error then there exists a high chances of more errors creeping into the system. As part of the error fixing, the developer finds it difficult to find the cause and fix the error to be removed from the system [21,22].

1.Error may be temporary in its nature and while debugging, the proper root cause is not found
2.Error may be caused by human which is more difficult to find out.
3.Error may be time consuming to be fixed and removed from the system.
4.Error gets fixed automatically while the developer fixes other errors in the program.
5.Error may be caused due to the logic used in the calculation like having Round-Off for the output number that is achieved.

The outline of the paper is presented here: Section 2 provides the proposed method. Section 3 deals with error bounds for normal densities. Section 4 and 5 provide the details about Chernoff and Bhattacharyya bound in the proposed method, respectively. Section 6 presents the algorithm for error probabilities and integrals of fault predictions. Section 7 concludes the entire work.

2 Methodology

The mathematical model is supported to solve the complex real-life model. The mathematical proof of the theory concept with logical solution comes from two problems, for large data sets have error analytics is very difficult part to identified the attributed of occurs, even though the number of tools available in the software industries. The error probability and error integral are also types of mathematical tools to find the probability of error and probability of correction. The error probability and error integral is deal in software data repositories in this research to identify the fault error in the software coding and development phases.

Bayesian decision theorem, is used to describe the conditional probability of an event and it is based on the prior knowledge of conditions that might be related to this event. The conditional probability removes the unwanted relevant event based on the conditional priority because the Beyers rule is applicable for less errors[23].

Probability of error: it makes the context of decision making of events, and it may be an making worm decision and it have a different type of error of its values [24]. The operation of a general classifier is called as Bayer's, or in other words it is refereed as the origin of its error. By considering the first two criteria of the Bayers i.e., by spitting the space into two segments, R1 and R2.

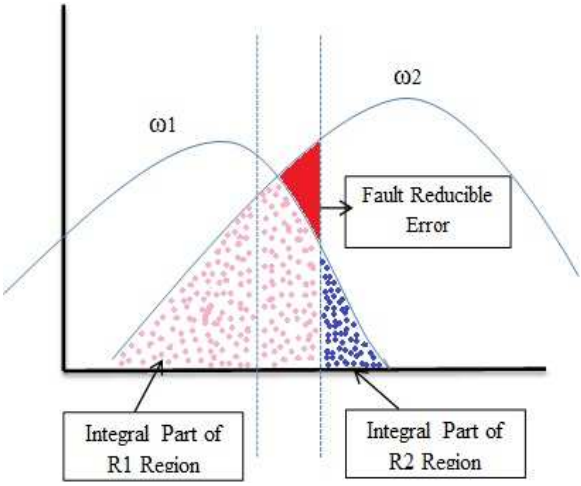

Fig. 1: Faults reducible identification region.

(with the reference of the probability of error).

Error can occur in two ways, and it occurs in the observation as follows (i) x falls in R2, which denotes state of nature as ω_1 and (ii) x falls in R1, which denotes true state of nature as ω_2. Since the following events are mutually coordinate and it can be exclusive and exhaustive, and the probability of error is calculated as:

$$\begin{aligned} P(error) &= P(x\varepsilon R_2, \omega_1) + P(x\varepsilon R_1, \omega_2) \\ &= P(x\varepsilon R_2|\omega_1)P(\omega_1) + P(x\varepsilon R_1|\omega_2)P(\omega_2) \\ &= \int_{\frac{R}{2}}^{\cdot} P(x|\omega_1)P(\omega_1)dx + \int_{\frac{R}{1}}^{\cdot} P(x|\omega_2)P(\omega_2)dx \end{aligned} \tag{1}$$

The two integral R_1 and R_2 represent the area in the tails of the function $P(x|\omega)P(\omega)$ since the judgment point of the regions R_1 and R_2 are chosen arbitrarily and the likelihood of error is not as small as it may be. Figure 1 shows the triangular area which is marked as fault reducible error, can be eliminated the decision boundary. In general, the bayes optimal make the decision boundary of regions which gives the lowest probability of fault error. The $P(x|\omega_1)P(\omega_1) \geq P(x|\omega_2)P(\omega_2)$ and it gives the advantages to classify R_1. The slighter quantity adds to the error integral; and it is exactly following the Bayes decision to rule achieve.

This result is described in the one-dimensional case as shown in Figure 1. The probability is flaws for equal priors and decision point is mentioned as non-optimal solution of the error. The blue colored area represents the probability of errors that decide ω_1 , when the state of nature is in fact ω_2 ; the yellow color represents the opposite, as given in the above equation 1. If the decision boundary is instead at the point of equal posterior probabilities, x_B, then this fault reducible error is left out and the total spotted area is the least probable. Bayes

decision provides the error rate by calculating through the above equation.

The other ways to be wrong than to be right, and is simpler to work out the prospect of being precise. The probability of the error rate can be correct by Equation (2).

$$P(correct) = \sum_{i=1}^{c} P(X \varepsilon R, \omega) \tag{2}$$

3 Error Bounds for Normal Densities

$$\sum_{i=1}^{c} P(x \varepsilon R, \omega) P(w) \tag{3}$$

$$\sum_{i=1}^{c} \int_{\frac{R}{2}} P(x \varepsilon R, \omega) P(w) dx \tag{4}$$

The general fault prediction, the result pretends neither on how the space is segregated into decision area R1 and R2 nor the form of the distributions shown in Equation (3) and Equation (4).
The Baye's classifier works by making the best use of probability to pick out the error in the region of R_1 and R_2. So that the integrand of the Equation (4) is utmost for all x; no other partition can yield a reduced possibility of error based on Equation (4).

The Baye's rule pledges the lowest average error rate of the fault prediction, and the result calculation on the limits for normal densities of the error limit. The Gaussian results would not specify the probability of error. The error for the Gaussian case would be fairly challenging naturally and especially in high dimensions, because of the irregular nature of the decision area R_1 and R_2in the integral.

4 Chernoff Bound

To device a bound for the error of the fault prediction is done by the following inequality:

$$min[a,b] \leq a^{\alpha} b^{1-\beta} for\ a,b \geq and\ 0 \leq \beta \leq 1 \tag{5}$$

By understanding Equation (5), the inequality can be assumed as $a > b$ in the condition as without loss of fault prediction and it noted that $b \leq a^{\beta} b^{1-\beta} = (a/b)^{\beta} b$ and this inequality is distinctly applicable, since $(a/b)^{\beta > 1}$. By Equation (4) and Equation (5), and apply this inequality to Equation (5) and the bounded region.

$$P(error) \leq P^{\beta}(\omega^1 P^{1-\beta}(2) \int P^{\beta}(x|\omega) P^{1-\beta}(x|\omega) dx \ \ for\ 0 \leq \beta \leq 1 \tag{6}$$

The integral is over all feature space and there is no restriction to impose combination limits corresponding to decision areas for the fault deduction region. If the conditional probability error is usual, the integral in Equation (6) can be estimated analytically, by docile the fault deduction in the region.

$$\int P(x|^1) P^{1-\beta}(x|\omega) dx = e^{-k(\beta)} \tag{7}$$

Where as the following equation predicts the error rate

$$K(\beta) = \frac{\beta(1-\beta)}{2}(\mu 2 - \mu 1)^t [\beta \sum 1 + (1-\beta)\sum 2]^{-1},\ \ (\mu 2 - \mu 1) + \frac{1}{2} ln \frac{|\beta \sum 1 + (1-\beta \sum 2|}{|\sum 1||\sum 2|^{1-}}. \tag{8}$$

Equation (8) shows a classic sample of $e^{-k(\beta)}$ the varies with β. The Chernoff bound on P(error) is set up by rationally or statistically deciding the value of β that decreases $e^{-k(\beta)}$, and substituting the results in Equation (8). The optimization has been done based on the Chernoff bound one-dimensional space, although the fact that the allocation among them might be in a space of arbitrarily high element.

5 Bhattacharyya Bound

The universal condition of Chernoff bound upon β which is more typical of extensive range of problems occurs by fault prediction. The bound is to loose extreme values ($\beta \implies 1\ and \beta \implies 0$), and tighten the transitional ones inside its boundary region. The optimal β value rests on the parameters of allocations and error probability. A computationally simpler but considerably fewer tight bound can be imitative by simply analyzing the consequences for $\beta = \frac{1}{2}$. Equation (9) is the called Bhattacharyya bound on the error.

$$P(error) \leq \sqrt{P}(\omega 1) P(\omega 2) \int \sqrt{P}(X\omega 1) P(X\omega 2) dx$$
$$= \sqrt{P}(\omega 1) P(\omega 2) e^{-k(1/2)} \tag{9}$$

Where by Equation (9), the way to have for the Gaussian case is:

$$K(1/2) = \frac{1}{8}(\mu 2 - \mu 1)[\frac{\sum 1 + \sum 2}{2}]^{-1}(\mu 2 - \mu 1) + \frac{1}{2} ln \frac{\frac{\sum 1 + \sum 2}{2}}{\sqrt{|\sum 1||\sum 2|}} \tag{10}$$

The Chernoff and Bhattacharyya limits are still adopted, even if the original distributions are not Gaussian. However, the distributions of fault predictions deviate from Gaussian and the limits do not cross its boundary limit.

6 Algorithms Error Probabilities and Integrals of Fault Predictions

The following algorithm shows the steps of software fault prediction using error probabilities and integral to identify

the fault boundary region with Mathematical equation.

Algorithms for P(Error) and P(Correction) of Faults Predictions

Step 1: Finding classification error

Condition 1: $R_2 \Leftarrow X \Leftarrow$True state of $\omega 1(P(X \sum R_2.\omega 1)$
Condition 2: $R_1 \Leftarrow X \Leftarrow$True state of $\omega 1(P(X \sum R_1.\omega 2)$
If, P(error) = integral of R_2 + integral of R_1

Step 2: Reducible error

Condition 1: X_B is eliminated if decision boundary is moved to region.
Condition 2: Least probability of error, If $P(X/\omega 1)P(\omega 1) > P(X/\omega 1)P(\omega 1)$
Condition 3: If the least quantity gives to the error integral; and it follows the Bayes decision rule.
Condition 4: Check if normal density (Step3) or outbound (Step 4) and store in Repository.
Condition 5: None of condition 4; to exit Step 2.

Step 3: Error of normal density for fault predication

Condition 1: Region error identification of R_1 andR_2.
Condition 2: If region R_1 is greater than region R_2, then the error intensity is high.
Condition 3: If region R_1 is equal to region R_2, then the lowest error is predicted.

Step 4: Error outbound among the integral region.

Condition 1: If the inequalities in the bound region then min [a, b] $\leq a^{\beta} b^{1-\beta}$, then the Fault prediction is of zero loss.
Condition 3: If the bounce is loose for great values for $(\beta \implies 1 and \beta \implies 0)$, and tighter for transitional ones in the boundary region.

The P (Error) and P (Correction), the data sets of values are stored in the software repository to eliminate the unwanted risk factors of fault prediction in coding part. The Gaussian helps at the level of the P(Error) that occurs in software development. Gaussian elimination is the best algorithm for solving the method of linear equations. This is used to understand the sequence of operation to perform the matrix of coefficients. The Gaussian helps to find the rank of matrix, to calculate the determinant of a matrix, and to calculate the inverse of an invertible square matrix to find the faults software repository.

7 Perspective

This paper states that the fault prediction has been analyzed and predicted by various algorithms and methods. In this research, the focus is on in-depth analysis of fault prediction by using of methodology of errors probabilities and integral method to predict the error occurring in the coding phases. The coding developed by the developers cannot identify the fault. The fault is identified through mathematical derivation through the boundary regions. The Chernoff bound and Bhattacharyya bound methods are supportive for the identification of faults prediction. These faults are used for grouping the faults and non-faults, which are saved in the software repository for the precautionary predictions. This is used when software developing life cycle is utilized by the testing team.

References

[1] Naheed Azeem and Shazia Usmani (2011), "Analysis of Data Mining Based Software Defect Prediction Techniques", Global Journal of Computer Science and Technology (GJCST), Vol.11, No.16, pp. 1-5, ISSN: 0975-4350.
[2] Wei Fu and Menzies T(2017), "Revisiting Unsupervised Learning for Defect Prediction",Proceedings of the 2017 & 11th Joint meeting on Foundations of Software Engineering (ESEC/FSE17),pp.1-12,DOI:10.1145/3106237.3106257.
[3] Bishop, C. M. (2006), "Pattern recognition and machine learning. Springer," ISBN 978-0-387-31073-2, 23.
[4] Hagan, M. T., Demuth, H. B., Beale, M. H., & De Jesus, O. (1996), "Neural network design," Vol. 20, Boston: Pws Pub.
[5] Ikeda, N., Watta, P., Artiklar, M., & Hassoun, M. H. (2001), "A two-level Hamming network for high performance associative memory," Neural Networks, 14(9), 1189-1200.
[6] M. Zidan, A.-H. Abdel-Aty, M. El-shafei, M. Feraig, Y. Al-Sbou, H. Eleuch, M. Abdel-Aty, "Quantum Classification Algorithm Based on Competitive Learning Neural Network and Entanglement Measure," Appl. Sci. 2019, 9, 1277.
[7] Bernardo, J. M., & Smith, A. F. (2009). Bayesian theory (Vol. 405). John Wiley & Sons, ISBN 0-471-92416-4. MR 1274699.
[8] A. Sagheer, M. Zidan, M. M. Abdelsamea, A Novel Autonomous Perceptron Model for Pattern Classification Applications, Entropy, 21(8), 763, 2019.
[9] Jifeng Xuan, He Jiang, Yan Hu, Zhilei Ren,Weiqin Zou, Zhhongxuan Luo, Xindong Wu (2015),"Towards Effective bug triage with software data reduction techniques", IEEE transaction on knowledge and data engineering,Vol. 27,No.1,pp.264-280.
[10] Viji, Rajkumar and Venkatesh (2017), "Hybrid Approach for Fault Prediction in Object-Oriented Systems", International Journal of Control Theory and Applications (IJCTA) by International Science Press, Vol.10,No.10,pp. 39-45.
[11] Gurtej Singh Ubhiand and Jaspreet Kaur Sahiwal (2017), " A Review on Software Mining: Current Trends and Methodologies," International Journal of Engineering Research and Application (IJERA), Vol.7, No. 4, pp. 40-45,ISSN:2248-9622.
[12] Julie Moeyersoms , Enric Junque de Fortuny, Karel Dejaeger , Bart Baesensand, David Martens(2015), "Comprehensible Software Fault and Effort Prediction: A Data Mining Approach", Journal of Systems and Software,v 100, pp. 80-90.

[13] Pon Periasamy A.R and Mishbahulhuda A (2017), "Data Mining Techniques in Software Defect Prediction," International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE),Vol.7, No.3, pp-300-303.
[14] Harald Altinger, Steffen Herbold, Friederike Schneemann, Jens Grabowski and Franz Wotawa (2017), "Performance Tuning for automotive software fault prediction," Software Analysis, Evolution and Reengineering(SANER),2017 IEE 24th International Conference, pp.526-530.
[15] Awni Hammouri, Mustafa Hammad, Mohammad Alnabhan, Fatima Alsarayrah (2018), "Software Bug Prediction using Machine Learning Approach" International Journal of Advanced Computer Science and Applications(IJACSA), Vol. 9, No. 2, pp. 78-83.
[16] Ishani Arora, Vivek Tearwal and Anju Saha(2015), "Open Issues in Software Defect Prediction," Proceeding of International conference on Information and Communication Technologies(ICICT 2014), v.46, pp. 906-912.www.sciencedirect.com.
[17] Rajdeep Kaur and Sumit Sharma (2018), "Various Techniques to Detect and Predict faults in software system: Survey", International Journal on Future Revolution in Computer Science and Communication Engineering,Vol.4,No.2,pp.330-336.ISSN:2454-4248.
[18] Zhiqiang Li, Xiao Yuan Jing and Xiaoke Zhu (2018), "Heterogeneous fault prediction with cost-sensitive domain adaptation," Journal of Software Testing, Verification and Reliability, Wiley Online Library,Vol.28,No.2,https://doi.org/10.1002/stvr.1685.
[19] Hand DJ (2007), Principles of data mining , Journal of Drug Safety," Vol. 30, pp. 621-622, ISSN: 0114-5916.
[20] Yubin Qu,Xiang Chen, Yingquan Zhao and Xiaolin Ju (2018), "Impact of Hyper Parameter Optimization for Cross-Project Software Defect Prediction," International Journal of Performability Engineeing, Vol. 14, No. 6,pp.1291-1299, DOI: 10.23940/ijpe.18.06.p20.12911299.
[21] Jinu M Sunil, Lov Kumar and Bhanu Murthy N L (2018), "Bayesian Logistic Regression for Software Defect Predication", Proceeding of 13th International Conference on Software Engineering (SEKE 2018), DOI:10.18293/seke2018-181.
[22] Kalaivani N and Beena R (2018), "Overview of Software Defect Prediction using Machine Learning Algorithms," International Journal of Pure and Applied Mathematics, Vol.118, No.20, pp.3863-3873. ISSN: 1314-3395.
[23] Bartomiej Wojcicki and Robert Dbrowski(2018), "Applying Machine Learning to Software Fault Prediction", In e-Informatics Software Engineering Journal, Vol. 12, No. 1, pp. 199-216, 2018.DOI: 10.5277/e-Inf180108.
[24] Chubato Wondaferaw Yohannese, Tianrui Li and Kamal Bashir(2018), "A Three-Stage Based Ensemble Learning for Improved Software Fault Prediction: An Empirical Comparative Study," International Journal of Computational Intelligence Systems,Vol.11,No.1,ISSN:1875-6883,doi:10.2991/ijcis.11.1.92.
[25] Hagan M, Demuth H and Beale M (1996), "Neural Network Design" USA: PWS Publishing company. 2nd Edition.
[26] Sarunas Raudys (1998), "Evolution and generalization of a single neurone: II. Complexity of statistical classifiers and sample size considerations," Journal of Neural Networks, Vol.11, No. 2, 31, pp. 297-313, https://doi.org/10.1016S0893-6080(97)00136-6.

S. KARUPPUSAMY is working as an Associate Professor in the Department of Computer Science and Engineering at Nandha Engineering College (autonomous) is located in Erode, Tamilnadu, India. He completes his B.Sc. in Computer Technology in the year 2003, and M.Sc. Computer Science in the Bharathiyar University in 2006 respectively, after he completed the M.E Computer Science and Engineering with first class in Anna University, Chennai. Currently, he is pursuing Ph.D. research in Software Engineering domain about the repository mining of fault prediction in software defects. He published research work outcome in various International conference and Journals.

G. SINGARAVEL received the D.C.T degree in Computer Technology in 1995 from Kongu Polytechnic College, and the Undergraduate degree B.E. (Electronics and communication and Engineering) in 1999 in Kongu Engineering College from the Bharathiyar University. He completed his master degree M.E. (Computer Science and Engineering) in 2002 at Arulmigu Kalasalingam College of Engineering from Madurai Kama raj University. He received his Ph.D. degree in the area of Information and Communication from Anna University, Chennai in 2010. He has 19 years of teaching experience. He is a Professor and the Head of the Department of Information Technology, K.S.R College of Engineering (Autonomous), Tiruchengode. He is recognized as a Supervisor for Ph. D Programme and M.Tech.(By Research) in Computer Science and Engineering for Anna University. He is trained certificate holder of High Impact Teaching Skills through Wipro Mission 10X and Trained Evaluator & Resource Person of NBA (National Board of Accreditation) by AICTE, New Delhi. He is in the editorial board member of 10 International/National Journals. He has published 90 papers in international, national journals and conference proceedings and having Google Scholar citations and h-index. He has guided 15 Master Degree projects. He is Guiding 16 PhD research scholars in various domains. His areas of research include Software Engineering, Agile Project Management, Green &Cloud Computing, Big Data Analytics, etc.,