International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

Best Suited Machine Learning Techniques for Software Fault Prediction

Devika S, Lekshmy P L

Abstract: ***In this world of emerging applications of software, it is always important to provide a quality assured product to customers. Software Fault Prediction popularly abbreviated as SFP is a major field which helps to provide quality assured products to customers. It helps to recognize modules that are bug-free and bug-prone in a software module. Machine learning techniques for both classification and determination are used for the purpose of software fault prediction. Software Fault Prediction is carried out prior to testing process without executing the source code, instead vital characteristics of software is taken into consideration. This early identification of faults can help software engineers to reduce the risk of system failure. A company does not always prefer to invest more expense on testing and in those situations, software fault prediction can have an upper hand in testing. The software fault prediction model will first train the learning techniques to generate base learners and then apply these base learners to unseen projects. It is always preferred to determine the count of faults rather than classifying each software module as fault-free and fault-prone. All software fault prediction techniques depend on base learners used and also nature of fault dataset. In this paper, the major learning techniques to determine software fault, characteristics of software fault dataset, etc. are discussed.***

Keywords: ***Software fault prediction, Decision Tree regression, linear regression, software fault dataset, repository, quality assurance.***

I. INTRODUCTION

Software Quality Assurance (SQA) is the process of developing first quality software or best quality product to customers. Quality assurance forms a major part in software fault prediction. The quality of a software can be measured by parameters such as error-proneness, integrity, flexibility, reliability and so on. Here the parameter discussed for software quality assurance is error-proneness. Software faults refers to those errors that affect the normal working of programs that is an unexpected result is got. The prediction of software fault is becoming complex as the size of source code increases. This plays an important role in software development process. And for this process previously reported fault related data is used. Crucial information about faults are- location of fault, number of faults, etc. These parameters can be used to enhance the software quality of later versions of software. A number of techniques are incorporated for this purpose. Some of them are decision tree, neural network, genetic programming, etc. Software release of previous version of software along with software metrics are used to build prediction models. The major two learning approaches used for the purpose are supervised learning and unsupervised learning. The approach that is discussed is supervised learning. It includes two phases – training phase and testing phase. Training phase outputs a model that can be applied to testing phase. If no testing data is available, then the input data is itself clustered. This technique is called unsupervised learning. The input data cannot be fully trusted as it is collected from companies and they may have errors. Different types of machine learning algorithms are used for this purpose. The algorithms discussed are decision tree, linear regression, random forest, multilayer perceptron, lasso regression, ridge regression, etc. To give better prediction analysis performance measures are calculated for each technique. Such a model if developed can help software developers to know the faults at an early stage. Here, the faults can be identified before handed, i.e., before the actual testing begins. Different performance measures like precision, support, recall, etc. were used to see how different datasets behave to the underlying model. Another problem majorly displayed is regarding showcasing the fact that different techniques find out different errors on different dataset. Bug, error and fault are used interchangeably in this paper.

II. LITERATURE REVIEW

Prediction techniques in machine learning for software fault prediction is one of the least investigated area. Yet it is an area to be investigated more since it is of great help for developers to know about the fault status before actual testing takes place.Schaffer [1] illustrated the approaches to the matter of classification, as well as ways for causing rule sets, call trees, Bayesian classifiers, and neural networks (back propagation). Two main conclusions drawn are listed. First, no single methodology or paradigm is uniformly superior. Second, problem-specific information will generally facilitate to guess that methodology can perform best. Here divides the information into 2 components, use one half as input to variety of classification algorithms, then select whichever rule produces the model most correct on the second half. On the positive side, cross-validation is used to pick out a classification methodology and might yield average predictive performance, well beyond what can be achieved with any other methodology.

Manuscript received on March 15, 2020.
Revised Manuscript received on March 24, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Devika S, department of Computer Science and Engineering, LBS Institute of Technology for Women.
Lekshmy P L, department of Computer Science and Engineering, LBS Institute of Technology for Women.

However, the poorer ways added, the upper the chance that one in every of them can seem superior to a higher strategy available. Taghi et al. [2] displayed a code metrics-based classification model.

It predicts a code module as either bug-free or buggy. This is often particularly necessary crucial for safety and mission-critical systems.

Here comparative analysis of three combined learners-bagging, boosting and logit-boost. The plan is to mix many classification accuracies by including expert opinion by compensating every model's weakness. A classifier is outlined as unstable if slight changes in the information produce considerably totally different classifiers. Samples of unstable classifiers generally include trees or neural nets. Whereas stable classification ways include Case- Based Reasoning (CBR). Overall, the simplest results were obtained by boosting. The combined weak learners might bring higher or similar classification accuracies than single or combined robust learners. Experiments showed that combined results were usually ready to improve individual classifiers. Also, it significantly reduces the time to pick out best model. However, the matter of optimum parameter choice isn't resolved here.

Ferenc et al. [3] worked with ASCII text file code that may be a form of program code within which ASCII text file is on the market to all or any within which the copyright holder grants users the rights to review, change, and distribute the code to anyone and for any purpose. Mozilla Firefox may be a free and ASCII text file application and is developed by the Mozilla Foundation. Bugzilla information containing dataset is employed for this purpose. It contains all bugs enclosed from starting of development of Mozilla. The following set of metrics used here are- WMC (Weighted ways per Class), dit (Depth of Inheritance Tree), LOC (Lines Of Code), LCOM (Lack of Cohesion on Methods- metric is about to zero whenever this answer ends up in a negative variety, LCOMN (Lack of Cohesion on ways permitting Negative value).

Two classifications are there-one class that contained categories while not bugs and therefore the alternative that contained categories with bugs (at least one). Four classes thought- about were: categories while not bugs, with just one bug, contained two to twelve bugs, and a class with thirteen or a lot of bugs. The shocking reality was that everyone yielded constant result. Out of all metrics used, LOC gave sensible result. however abundant exactness wasn't recognized.

Elish et al. [4] discussed with distinguishing faulty categories in object-oriented code is a challenge for code developers before the code is discharged. Ensemble classification techniques have received abundant attention and have incontestable promising capabilities in classification accuracy over single classifiers. However, these techniques haven't been applied and evaluated within the context of distinguishing faulty categories in object-oriented code. The objective of this study is to work out the extent to that bagging and boosting ensemble techniques provide a rise in classification accuracy over single classifiers. First, apply and compare six common and customary process intelligence models in distinguishing faulty categories. Second, apply and evaluate the impact of 2 common ensemble techniques (bagging and boosting) in the classification accuracy over single classifiers. The results indicate that ensemble techniques like bagging and boosting gave an improved accuracy over most of the present and investigated single classifiers. The matter of optimal parameter choice and stopping criteria for ensemble size isn't solved.

Hansson et al. [5] considered those modules that are defect prone and those which require intensive testing. Two ways are used for defect prediction- methodology using machine learning. Adoption of learning techniques in Software Defect Prediction is due to lack of understanding the state of affairs. Technology Acceptance Model and Technology Adoption Framework is that the major thought that enforced learning techniques to action. The Technology Acceptance Model (TAM) is acceptance mode data systems theory that tells however users come back to just accept and use a technology. The Technology Adoption Framework is employed for the assessment of however folks build choices relating to new technology adoption. Various styles of Machine Learning (ML) techniques used for code defect prediction are Trees (Decision Trees), Support Vector Machines (SVMs), Artificial Neural Networks (ANNs) and Bayesian Belief Networks (BNNs). A Bayesian network is a graphical model that represents a collection of attributes and their conditional probabilities with the assistance of a directed acyclic graph. All the mentioned techniques don't need any domain information. this can successively facilitate corporations, researchers and power vendors to fulfill the precise data wants. However, this method wasn't ready to verify the simplest attributes.

Arya et al. [6] illustrated various data mining methods used for software defect predictions. Some of them are named below. They are regression, clustering, classification, etc. Some of the classification techniques used for the same are Neural Network, Naive Bayesian, Decision Tree, Support Vector Machine, Case Based Reasoning, etc. Here various classification techniques such as Supervised, Un- supervised and Semi-supervised, are applied on various datasets based on existing software metrics. The proposed method can also be used as an automated tool to predict defects using very little time for project managers, software developers and testers. Major disadvantage with this approach is that there is no mechanism for comparing the results of supervised classification techniques on different datasets and open source projects to analyze the best classification technique to predict the defect so that a first quality software product can be carried out.

David Bowes et al. [7] displayed defect prediction models that is used to directly test defect-prone code. Earlier defect prediction models consist of four main modules. First, the model uses independent variables. Second, the model depends on a specific modelling technique. Third, dependent variables are produced by the model. Fourth, a method is designed to measure the predictive performance of a model. The main goal of this paper is to identify classification techniques which predicts software fault in every software module. Analysis shows that the classifier's inconsistency occurs in a variety of different software domains, which includes both open source and commercial projects. Performance measures displayed that defect prediction models are performing similarly.

But even when similar performance is exhibited, different defects are identified by different classifiers. This has important implications for defect prediction. First, assessing predictive performance using conventional performance measures such as f-measure, precision or recall gives only a basic picture of the performance of model. As a result, more precise performance measures should be used.

Yang et al. [8] illustrated arranging software modules in order of defect count, known as software defect prediction. It can help developers and testers to focus on software modules that are defect-prone and quickly identify those modules. The process mainly includes two parts. They are data and model construction methods. First, obtain the data from software modules, according to software metrics (also referred to as features or attributes), such as lines of code and previous defects. Second, modeling approaches such as linear regression (LR) and random forest (RF) are employed to construct a model based on the data from modules with known defect numbers. Finally, the constructed model is used to predict the defect information of the software modules. Therefore, an attempt to a model that has the merits of generalized LR models and give better performance when the multicollinearity problems exist. As a result, two latest regression methods - ridge regression (RR) and least absolute shrinkage and selection operator (lasso regression) are used for sorting software modules in order of defect count.

The choice of parameters of biased estimation methods for software defect prediction and compare them with more methods is absent in this mechanism.

Kumar et al. [9] illustrated the problem of difference in performance, which can be solved by using a new approach. This method partitions the input dataset into different subsets. Then it trains learning techniques for each subset, and amalgamate the outcomes of all the learning techniques. The approach mainly focusses on dynamic selection of learning techniques to predict the number of software faults. For a given unseen module, it first locates its adjacent neighbor that is similar to testing module. Then it chooses the best learning technique that is one with minimum errors in the region of that module subset. Presented approach selects the best learning technique (Linear regression, Multilayer perceptron, Decision tree) for each unseen testing module in the given testing dataset. Only few learning techniques are considered and also datasets of different domains are not taken into consideration.

Kavitha et al. [10] presented a strategy with regression test prioritization techniques. Test case prioritization techniques as the name suggests, is used to rank test cases for execution so that those with higher priority, according to some criterion are executed earlier than those with lower priority. Assigning priority to test cases is carried out by considering two factors. They are rate of fault detection and fault impact. Testers provide rank to the test cases based on above critiria so that those test cases with the highest priority according to some criterion are run first. This activity of prioritization technique does not discard test cases. The results indicated that the proposed technique lead to improved rate of detection of severe faults in comparison to random ordering of test cases. The results showed that the proposed prioritization technique is much more effective than existing techniques. The problem is that there is no test case prioritization over requirement analysis on demand which must be corrected.

Afzal et al. [13] proposed a fault detection model using Genetic programming. He constructed and performed fault prediction models for software systems. The number of fault counts are used as the independent variable. The outcomes of experiments showed that Genetic Programming produced a significant performance for predicting software fault counts.

Khoshgoftaar et al. [14] performed a comparison of different count models for their use in predicting number of faults. It included five different count model-based techniques. This evaluation model has been done using two different software systems containing various software metrics related to complexity measures and object-oriented measures.

Grave et al. [16] attempted to evaluate change history metrics for predicting software faults using a linear regression (LR) technique. He performed experiments for a large telecommunication system. The results showed that change history metrics produced a better performance when compared to the other metrics for predicting software faults.

Ostrand et al. [17] studied different methodologies using a negative binomial regression (NBR) technique to determine the number of faults in the given software module prior to testing. He found that NBR demonstrated an accurate performance when predicting the number of software faults.

Rathore et al. [18] introduced the concept of predicting the number of faults in software rather than performing binary classification on faults (i.e., faulty or non-faulty) with the help of characteristics of software. Even if the accuracy of binary classification was found excellent, but number of faults cannot be predicted using binary classification technique. Neural network and genetic programming were used for prediction of number of faults. Later a comparison between the two techniques was carried out. And for this purpose, the prediction model was implemented over ten fault datasets from Promise data repository. One of the major conclusions drawn from the above experiments where that neural network produced better result for smaller datasets, while genetic programming produced better for larger datasets. But genetic programming produced better result while taking into consideration performance measures like recall, etc. Performance measures used in this model are recall, error rate and completeness analysis.

Many regression approaches have been used for software fault prediction. But no clear winner was seen among them. Another constrain to be considered was that depending on the input dataset, the output also differed. It depended not only on the dataset used but also to an extend the learning technique used. The introduction of ensemble methods must also be considered as it worked better than single methods because always a group opinion is considered to be best. Another important consideration is in object-oriented software prediction where object-oriented software metrics are used. For this purpose, logistic regression model is used. Research studies have searched the use of computational intelligence models to predict fault proneness class as regression problems. Artificial Neural Networks is introduced as a new approach for classification of class faults in object-oriented software.

Multilayer perceptron was used to identify faulty classes. Some authors illustrated empirical analysis of software fault using Bayesian methods such as naive bayes classifier, etc.

III. SOFTWARE FAULT DATASET

For the purpose of software fault prediction, it is preferred to learn about the information corresponding to the software system and it is termed as software metrics. The software metrics must include fault value for prediction. Collecting fault data by self is not advised as one can never verify the correctness. Thus, researchers used the open source or publicly available datasets. Software fault dataset consists of information such as- software metrics, fault information (. i.e., either faulty or non-faulty / number of faults).

Software fault prediction techniques involves use of repository for the development of software. Some of them used are source code repository, bug repository and archived communication repository [19]. Source code repository is used to track the changes made in source code during development e.g. Git. Bug repository keeps track of all issues reported by developers or users e.g. Bugzilla. Archived communication repositories include chats among various developers related to software development and this in turn can be used by managers for better view about software under construction. Software metrics can be calculated using automated tools such as Source Monitor (for C, C++, etc.) and Understand (for Java). All the bugs identified by bug repositories have to be linked with corresponding software module using automated tools or by pattern matching. Fault dataset repositories are classified into three categories – private dataset repository, partially public dataset repository and public dataset repository. Public data repositories like Nasa and promise data repositories can be used for this purpose. Also, one cannot expect all datasets to have all software information. Some have number of faults, others have details like faulty or non-faulty modules. The bug or fault prediction datasets like PROMISE [20], ECLIPSE, NASA data repository will be used for this purpose. It's a group of models and metrics of software package systems and their histories. The datasets may or may not have related attributes. The goal of such a dataset is to permit individuals to check totally different bug prediction approaches and to judge whether or not a brand- new technique is an improvement over existing one. Now let us see attribute information about traditional metrics. Traditional metrics were those used at early days of software development. Some of them are -loc (line count of code), v(g) (cyclomatic complexity), n (total operators + operands), etc. After a considerable study, it was evident that object-oriented metrics out performed traditional metrics. Object oriented (OO) metrics are those metrics that deals with object-oriented technology developed software. Some of the OO metrics are- Weighted method count (WMC), which refers to the no. of member functions and operators defined in each class, Depth of Inheritance (DIT), which refers to the no. of ancestors of class, etc.

IV. APPROACHES USED

Artificial Neural Network as discussed in [11] is an information processing system which resembles in characteristics with a biological neural network. All neurons are interconnected to each other with help of a connection link. These are associated with weights and information, in turn can be used to solve a particular problem. The Sigmoid activation function is the most frequently used activation function in neural nets. Thus, the models of ANN are characterized by the three basic concepts namely: 1) The models synaptic interconnections; 2) The training or learning rules adopted for updating and adjusting the connection weights; 3) Their activation functions. The activation function used is rectified linear unit function. The ReLU activation function's output ranges from-1 to 1 and is better suited for the normalized input. It is implemented using weka datamining tool.

Random Forest mentioned in [7] is an ensemble learning technique for classification and regression. It consists of a set or ensemble of straightforward tree predictors, where every technique is capable of manufacturing a response once given with a collection of predictor values. For classification issues, this response takes the shape of a category membership, that associates, or classifies, a collection of predictor values with one in every of the classes within the variable quantity or else, for regression issues, the tree response is an estimate of the variable quantity given the predictors. The Random Forest rule was developed by Breiman.

Bagging as mentioned in [2] is an ensemble learning technique. It is otherwise known as Bootstrap Aggregating. The idea plan of bagging is to construct every member of the ensemble from a unique dataset, and to predict amalgamation by either uniform averaging or balloting over category labels [3]. A bootstrap samples N thing uniformly indiscriminately with replacement. which means every classifier is trained on a sample of examples considered with a replacement from the data set, and every sample size is adequate to the dimensions of the first data set. Therefore, bagging produces a combined model that usually performs higher than the only model created from the first single data set.

Boosting mentioned in [2] is another ancient ensemble methodology, and Adaboost is that the foremost well-known of the Boosting family of algorithms that trains models consecutive, with a newest model trained at every case. Adaboost constructs associate ensemble by acting multiple iterations. Throughout the course of methodology for every iteration it uses entirely altogether entirely completely entirely completely different example weights. the burden of incorrectly classified examples unit of live accrued, as a result it ensures misclassification errors for these examples count tons any heavily among succeeding iterations. This procedure provides a series of classifiers that complement each other, then the classifiers unit of live combined selectively. Another technique, regression performs the task to predict a quantity worth (y) supported a given variable (x). So, this prediction technique finds out a linear relationship between input and output variables. Hence, the name is straightforward regression. If we have got a bent to tend to tend to plot the variable (x) on the axis and variable (y) on the axis, regression offers everyone a line that with regards to all or any or any or any closely fits the knowledge points.

Adaboost constructs associate ensemble by acting multiple iterations. Throughout the course of methodology for every iteration it uses entirely altogether entirely completely entirely completely different example weights.

the burden of incorrectly classified examples unit of live accrued, as a result it ensures misclassification errors for these examples count tons any heavily among succeeding iterations. This procedure provides a series of classifiers that complement each other, then the classifiers unit of live combined selectively. Another technique, regression performs the task to predict a quantity
worth (y) supported a given variable (x). So, this prediction technique finds out a linear relationship between input and output variables. Hence, the name is straightforward regression.

If we have got a bent to tend to tend to plot the variable (x) on the axis and variable (y) on the axis, regression offers everyone a line that with regards to all or any or any or any closely fits the knowledge points.

Decision tree as mentioned in [15] refers to a fateful model that uses the item's observation as branches to know the item's target worth among the leaf. Decision Tree is as well a tree with decision nodes, that have over one branch and leaf nodes, that represent the choice or decision to be made after splitting. Simple CART is a variation of decision tree.

Genetic programming as discussed in [18] is a search-based algorithm which finds out the best solution to perform a given task. With the help of a fitness function, the survived individuals are selected and is proceeded for the next iteration. The fitness function used for fault prediction is the square root of the difference between the obtained values and actual values of the number of faults in all fitness cases. After the convergence condition is reached, the best individual selected as the optimal solution. A GPLAB 3.0 (Genetic Programming Toolbox for MATLAB) toolbox is used to implement genetic programming in MATLAB.

V. DISCUSSION AND RESULT

After a good study in the field of software fault prediction, the importance of applying learning techniques to partitioned datasets rather than doing the same without partitioning is discussed. This solves the problem of detecting different techniques detected different errors when working with different datasets. Overview of basic software fault prediction technique is as shown in Fig. 1, given below. refuse to work with software fault prediction due to lack of knowledge in this field.

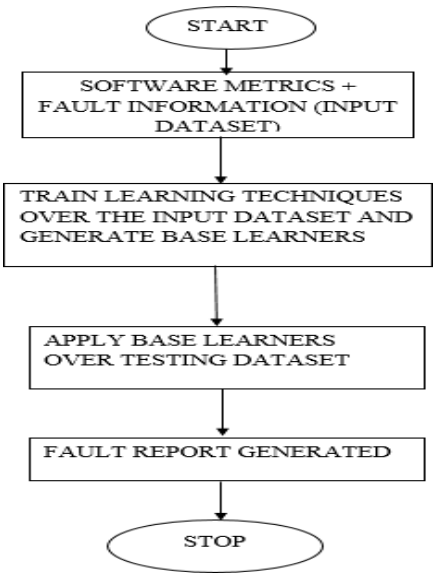

Fig. 1 Overview of basic software fault prediction

First, we divide software fault dataset into training data and testing dataset. The learning techniques are applied to training dataset to generate base learners. The generated base learners are then applied to testing dataset and prediction is carried out. The generated fault report can either provide a binary classification of faults or it may predict number of faults. And this strongly depends on our requirement. A quick identification of fault-proneness modules as a result of prediction techniques can improve the efficiency of developing and testing phase. Software fault prediction helps to identify the faults without executing the code, i.e., without testing the code by testers. But still software companies prefer their mode of working. Now let us see a comparative study on types of software fault prediction. They are as given below in Table I [19].

Table I. Comparative study between types of software fault prediction techniques

Binary classification of software faults	**Prediction of number of faults**
When a software module is identified with one or more number of faults, it is said to be faulty. Otherwise, it is said to be non- faulty.	It predicts the number of faults on each software module. It is always preferred to predict rather than classification.
Confusion matrix- based performance evaluation measures such as accuracy, precision, recall, etc. are used here.	Generally, error- based performance measures such as mean absolute error and mean relative error are used here.
Even if a lot of works have been carried out in this regard, still doesn't converge to any generalized statement.	Most works are based on predicting the number of faults in post-release with the help of pre- release software.
An accuracy of 70% - 90% was obtained.	An accuracy of 90% and above was obtained.

As far as discussed in section IV, decision tree is said to outperform all other mentioned techniques. This is exhibited due to the easy, user friendly and transparent nature of decision tree. In most of the cases Simple CART is preferred. Table II given below shows the final observations by some of the techniques discussed in section IV [4] [18] [19]. For both the ensemble techniques, random forest is taken as base learner with an ensemble size of 30.

RESULTS

Table II. Final Observations

Sl. No.	Category	Techniques Used	Remarks
1.	Tree- based techniques	Simple CART	Avg acc- 93.15%
2.	Tree- based techniques	Random Forest	Avg acc -91.15%
2.	Neural Network	Genetic Programming	recall values 25% ~ 45%
3.	Neural Network	Neural Network	recall values 6% ~ 30%
4.	Ensemble Techniques	Bagging	Acc – 72.41%
5.	Ensemble Techniques	Boosting	Acc - 74.48%

VI. CONCLUSION

Software fault prediction is a rarely discussed area in software engineering which largely helps software industry to detect the faults at once. Major conclusions that can be drawn after this study are - software fault prediction enables test cost reduction, early fault detection in software system, quality software production. The importance of implementing software fault prediction model before the actual testing is already discussed and it in turn helps to reduce the cost. A conclusion to the above statement was reached since the faulty modules identified by both testing phase and software prediction models where the same. Some of the issues such as imbalance of software modules, high data dimensionalities, noisy data, etc. must be dealt carefully. The functioning of Software Fault Prediction process highly depends on the use of base learners and uniqueness of software fault datasets. The above two challenges raise the need of using multiple learning techniques for predicting faults in a given software system. According to previous studies decision tree regression gives a considerable accuracy in predicting software faults. Even if the techniques discussed in section IV exhibit similar predictive nature, there is a major difference in the accuracy obtained.

REFERENCES

1. Schaffer, Cullen. "Selecting a classification method by cross-validation." Machine Learning 13.1 (1993): 135-143.
2. Khoshgoftaar, Taghi M., Erik Geleyn, and Laruent Nguyen. "Empirical case studies of combining software quality classification models." Third International Conference on Quality Software, 2003. Proceedings.. IEEE, 2003.
3. Gyimothy, Tibor, Rudolf Ferenc, and Istvan Siket. "Empirical validation of object-oriented metrics on open source software for fault prediction." IEEE Transactions on Software engineering 31.10 (2005): 897- 910.
4. Aljamaan, Hamoud I., and Mahmoud O. Elish. "An empirical study of bagging and boosting ensembles for identifying faulty classes in object- oriented software." 2009 IEEE Symposium on Computational Intelligence and Data Mining. IEEE, 2009.
5. Rana, Rakesh, et al. "A framework for adoption of machine learning in industry for software defect prediction." 2014 9th International Conference on Software Engineering and Applications (ICSOFT-EA). IEEE, 2014.
6. Prasad, M. C., Lilly Florence, and Arti Arya. "A study on software metrics based software defect prediction using data mining and machine learning techniques." International Journal of Database Theory and Application 8.3 (2015): 179-190.
7. Bowes, David, Tracy Hall, and Jean Petrić. "Software defect prediction: do different classifiers find the same defects?." Software Quality Journal 26.2 (2018): 525-552.
8. Yang, Xiaoxing, and Wushao Wen. "Ridge and lasso regression models for cross-version defect prediction." IEEE Transactions on Reliability 67.3 (2018): 885-896.
9. Rathore, Santosh Singh, and Sandeep Kumar. "An Approach for the Prediction of Number of Software Faults Based on the Dynamic Selection of Learning Techniques." IEEE Transactions on Reliability 68.1 (2018): 216- 236.
10. Kavitha, R., and N. Sureshkumar. "Test case prioritization for regression testing based on severity of fault." International Journal on Computer Science and Engineering 2.5 (2010): 1462-1466.
11. Mundada, Devendra, et al. "Software fault prediction using artificial neural network and Resilient Back Propagation." International Journal of Computer Science Engineering 5.03 (2016).
12. Krizhevsky, A., and I. Sutskever. "ImageNet classification with deep convolutional neural network." Communications of the ACM 60.6: 84ą-90.
13. Afzal, Wasif, Richard Torkar, and Robert Feldt. "Prediction of fault count data using genetic programming." 2008 IEEE International Multitopic Conference. IEEE, 2008.
14. Gao, Kehan, and Taghi M. Khoshgoftaar. "A comprehensive empirical study of count models for software fault prediction." IEEE Transactions on Reliability 56.2 (2007): 223-236.
15. Hammouri, Awni, et al. "Software bug prediction using machine learning approach." International Journal of Advanced Computer Science and Applications 9.2 (2018): 78-83.
16. Graves, Todd L., et al. "Predicting fault incidence using software change history." IEEE Transactions on software engineering 26.7 (2000): 653-661.
17. Ostrand, Thomas J., Elaine J. Weyuker, and Robert M. Bell. "Where the bugs are." ACM SIGSOFT Software Engineering Notes 29.4 (2004): 86-96.
18. Rathore, Santosh Singh, and Sandeep Kuamr. "Comparative analysis of neural network and genetic programming for number of software faults prediction." 2015 National Conference on Recent Advances in Electronics & Computer Engineering (RAECE). IEEE, 2015.
19. Kumar, Sandeep, and Santosh Singh Rathore. Software Fault Prediction: A Road Map. Springer Singapore, 2018.
20. http://www.cc.uah.es/drg/c/RHH_RAISE12_Repos.html

AUTHORS PROFILE

Devika S is pursuing (4th Semester) Master's Degree in Computer Science and Engineering from LBS Institute of Technology for Women, Kerala, India affiliated under Kerala Technical University. She received her Bachelor's degree in Computer Science and Engineering from Mohandas College of Engineering and Technology, Kerala, India in the year 2018, affiliated under Kerala University. She developed a software to identify plagiarism on multithreaded programs and also developed a software that would enable intra college communication.

Lekshmy P L received her Bachelor's degree in Information Technology from M S University, Tamil Nadu in 2004. Then she obtained her Master's degree in Computer Science and Engineering from Karunya Deemed University, Coimbatore in 2006. Currently, she is working as an Assistant Professor in Computer Science and Engineering, L B S Institute of Technology for Women, Trivandrum, University of Kerala (since 2008). Her current research interests are Privacy Preserving Datamining and Big Data analytics.