ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/326074736

A Literature Review Study of Software Defect Prediction using Machine Learning Techniques

Article · June 2018

DOI: 10.23956/ijermt.v6i6.286

CITATIONS
17

READS
2,552

3 authors, including:

Feidu Akmel
Wolkite University, Ethiopia
4 PUBLICATIONS **51** CITATIONS

SEE PROFILE

Ermiyas Birihanu
Wolkite University, Ethiopia
15 PUBLICATIONS **62** CITATIONS

SEE PROFILE

All content following this page was uploaded by Feidu Akmel on 09 September 2019.

The user has requested enhancement of the downloaded file.

International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-6, Issue-6)

Research Article | **June 2017**

A Literature Review Study of Software Defect Prediction using Machine Learning Techniques

Feidu Akmel[*]
(MSc.) Computer Science, Wolkite University, Wolkite, Ethiopia

Ermiyas Birihanu
(MSc.) Software Engineering, Wolkite University, Wolkite, Ethiopia

Bahir Siraj
(MSc. Candidate) Addis Ababa Science and Technology University, Ethiopia

Abstract-

Software systems are any software product or applications that support business domains such as Manufacturing,Aviation, Health care, insurance and so on.Software quality is a means of measuring how software is designed and how well the software conforms to that design. Some of the variables that we are looking for software quality are Correctness, Product quality, Scalability, Completeness and Absence of bugs, However the quality standard that was used from one organization is different from other for this reason it is better to apply the software metrics to measure the quality of software. Attributes that we gathered from source code through software metrics can be an input for software defect predictor. Software defect are an error that are introduced by software developer and stakeholders. Finally, in this study we discovered the application of machine learning on software defect that we gathered from the previous research works.

Keywords: Machine Learning, Software Defect, Software Engineering, Machine Learning Techniques

I. INTRODUCTION

When there repeatedly exists a software failure in system through time it automatically leads to software defect. Software defect are an error that are introduced by software developer and stakeholders. The main objective of software defect prediction is to improve the quality, minimized cost and time of software products. Software defect is also referred to as bug can be defined as shortage in the software product that causes the software not to perform its task as the programmer and customer needed.

Machine Learning is one of the most vital and motivating area of research with the objective of finding meaningful information from huge data sets. The basic purpose of machine learning is to extract useful pattern from the data, mining data may be structured format (example. multiple data base) or text mining: unstructured data (example, natural language document).

In this study, we deeply observed the major factor of software failures that lead the software company to software defect, consume cost and times to test and maintenance after delivered to the stakeholders. In addition, we examine the recommended solutions to software failures, machine learning concepts and application of machine learning on software engineering, specifically for software testing and maintenances.

The rest of this paper is organized as follows:**Section 2** explains major factor for software failures and the recommendations filter out by the researcher;**Section 3** summarized common Known defect predictors; **Section 4** discussed about machine learning concepts and the application area of machine learning with regarding of software engineering; specifically, for software defects.**Finally**, we tried to assess the research works with respect to machine learning on software defect and classified it based on methods they used namely based on classification method, Clustering method and ensemble methods. The researchers conclude this paper with a summery and provide the future direction for researchers.

II. MAJOR FACTOR OF SOFTWARE FAILURE

Software systems are any software product or applications that support business domains [1], such as manufacturing, banking, healthcare, insurance, aviation, social networking, e-commerce or any other domains. In order to develop and design software system, there is a need of finance, human experts in domain area, a lot of time, tools and infrastructures. Now a day, even if the software company has a lot of experience in designing and developing projects, (according to Table 2) the software failure is increasing from time to time, which leads to loss of capital, time and energy consumption. The system may fail due to fault happened every software development cycle or customer may not provide you the exact requirement because they do not have awareness of information technology projects or political, cultural issues.

The survey participants were also asked about the factors that cause projects to be challenged. The most common known cause of software failures is Lack of user involvement[1][2][3],unclear goals and objectives[3][4], incomplete requirement specification[3][4], lack of resources[3], inadequate project planning and scheduling [3-5], Poor communication between Team members[3][4] and Poor Testing. From Table-1 below, we have been looking that lack of user input and incomplete user requirement specification are the major factors for projects to be successful. The major factors for software failure are depicted in Table I below.

© www.ermt.net All Rights Reserved

Table I Major factor for software failures[6]

Project Challenged Factors	% of Responses
1. Lack of User Input	12.8%
2. Incomplete Requirements & Specifications	12.3%
3. Changing Requirements & Specifications	11.8%
4. Lack of Executive Support	7.5%
5. Technology Incompetence	7.0%
6. Lack of Resources	6.4%
7. Unrealistic Expectations	5.9%
8. Unclear Objectives	5.3%
9. Unrealistic Time Frames	4.3%
10. New Technology	3.7%
Other	23.0%

According to CHAOS MANIFESTO in the 2013 report, they were doing the survey from 2004 to 2012 software projects and categorizing it based on the percentage of failure, success and challenging projects as shown in Table II below.

Table II Percentage detail from 2004 – 2012**[7]**

	2004	**2006**	**2008**	**2010**	**2012**
Successful	29%	35%	32%	37%	39%
Failed	18%	19%	24%	21%	18%
Challenged	53%	46%	44%	42%	43%

A. Recommendation to Address Software Failures

As we have discussed section II, we mentioned the cause of software failures, however, any trouble has a solution here we list out some of the recommendations for success the projects.

- Monitor the project - the accuracy of the estimate.
- Fully satisfied the user requirements.
- Do not depend on a single cost or schedule estimate.
- Put the clear objectives and goal.

III. SOFTWARE DEFECT PREDICTOR

The software defect predictor is the way or a method that helps for software testing and software development life cycles. Software defect is also referred to as bug can be defined as shortage in the software product that causes the software not to perform its task as the programmer and customer needed. There are different types of classification for software defects, according to the Institute of Electrical and Electronics Engineers (IEEE) standard 104[8] which are:

- **Defect:** The lack to perform the tasks and system requirements that are provided by developer and customers.
- **Error:** This can be happening by customer due to the knowledge about the deficiency the software, produce incorrect results
- **Failure:** Stop its previous required function of the software products or incorrect result will be provided for each input that is given by the customers.
- **Fault:** it is clear or obvious error occurred in the software products.

Depending on the above software defect classifications, a defect that is occurring in software product lead malfunctioning and health problems in case of serious and critical software, such as NASA software products for Space sciences.

IV. MACHINE LEARNING CONCEPT

One of the main differences between how people and computer work is that human, performing any kind of activity usually simultaneously expand efforts to improve the way they perform it. This is to say, human can perform any tasks with the versatility means while not machines.

Machine learning algorithms ‘learn’ to predict outputs based on previous examples of relationships between input data and outputs. The models that are constructed based on the relationship between input and output slowly improved by testing and its predictions and correcting when wrong[9]; according to Tom Mitchell definition, the machine learns with respect to particular tasks **T,** performance **P** and Experience **E**[10].

© www.ermt.net All Rights Reserved

A. Machine Learning Tasks

A system that is used to understand the concept and its environment using a simplified interpretation of the environment using model is called cognitive system [11]. The step that we pass to construct the model is known as inductive learning. The Cognitive system is able to combine its experience by constructing new structures is patterns. The constructed model and pattern by cognitive system is called machine learning.

Models that are described as predictive since it can be used to predict the output of a function (target function) for a given value in the function's domain while informative pattern are characterized only describes the portion of data.

Machine learning task classified into four that are supervised, unsupervised, semi-supervised and reinforcement learning;however, the most known task is supervised and un-supervised learning [8][12][12]. In section a and section b below, we looked some of supervised and un-supervised learning below.

i. Supervised Learning

It is the machine learning task of inferring a function from labeled training data. The training data consists of a set of training examples. In supervised learning, each example is a pair consisting of an input object and a desired output value. It has two known supervised learning tasks, classification and regression. Classification concerns on building predictive model for function with discrete range, while regression concern on continuous range model building. A lot of researchers in machine learning are focused on supervised learning[13].

The most common supervised machine learning methods includes concept learning, classification, rule learning, instance based learning, Bayesian learning, liner regression, neural network and support vector machine[8][11][14].

ii. Unsupervised Learning

This is also called learning from observation. In unsupervised learning the system has to explore any patterns based only on the common properties of the example without knowing how many or even if there are any patterns. The most common method in unsupervised learning's is association rule mining, sequential pattern mining and clustering.

B. Machine Learning in Software Engineering

The Machine learning is not a difficult science [15]. Machine learns automatically from training data and it generate summaries of data or existing systems in a smaller form while the software engineer could able to use machines in order to minimize the system development phases time and cost consumptions.

One thing that overcomes machine learning integrating with software engineering is developing machine learning based solution to software engineering problems [11]. The data and the complexity of pattern in software engineering needs pre-processed before applying machine learning method, this is similar with other applications.

Component-based approach fault prevention methods have tremendous attentions from developers now a time, which reduce the development cost and time as well as to improve the quality and reliability of the projects by breaking the project in to separate components, however it is only practical for finding the problems in the quality of individual components [16]. One of the solution to resolve the problem is through machine learning approach to software engineering.

In order to estimate the quality of software we used an attribute such as software development effort, software reliability and productivity of programmer to predict the important of model in software engineering. Early software quality prediction to enhance the performance of system via machine learning techniques (case based reasoning and fuzzy logic) was studied [17].

According to researches, Software engineering problems that are ready to resolved by machine learning, which are software measurement selection, defect prediction models, software reuse qualification, software requirements gathering, software quality estimation, project management, software testing[9][18][19][20][21], we also called it an application of machine learning in software engineering. Among software engineering problems that we listed, researcher chooses software defect prediction via machine learning methods to conduct this study.

In Table III we were looking machine learning approaches (methods) for software engineering tasks.

Table III SE (software engineering) tasks and applicable ML methods[9]

SE tasks	**Applicable Type(s) of Learning methods**
Requirement engineering	AL, BBN, LL, DT, ILP
Rapid prototyping	GP
Component reuse	IBL (CBR4)
Cost/effort prediction	IBL (CBR), DT, BBN, ANN
Defect prediction	BBN
Test oracle generation	AL (EBL5)
Test data adequacy	CL
Validation	AL
Reverse engineering	CL

© www.ermt.net All Rights Reserved

C. Machine Learning in Software Defect Prediction

The field of machine learning has been growing rapidly, producing a variety of learning algorithms for different applications. As we discussed in section 2.5.4.1 one of the application of machine learning is software engineering. The ultimate value of those algorithms is to a great extent judged by their success in solving real-world problems. Therefore, algorithm reproduction and application to new tasks are crucial to the progress of the field. However, various machine learning researchers currently publish for software fault prediction model development. Now, we categorizing successful software defect model into three that are based on classification, clustering and ensemble methods.

V. BASED ON CLASSIFICATION METHODS

EzgiErturk et al. [22] the data set for the experiment are collected from the PROMISE Software Engineering Repository and applied McCabe software metrics. The algorithm they are using form experiment was SVM, ANN and ANFIS (new adaptive model proposed), the performance measure was 0.7795, 0.8685, and 0.8573 respectively.

Another successful paper was published by Surndha Naidu et al.[23], the primary goal of this paper was finding the total number of defects in order to minimize time and cost. The defect was classified into five parameters such as Volume, Program length, Difficulty, Effort and Time Estimator. They were using ID3 classification algorithm, to classify defects.

Malkit Singh et al. [52] Explored the fault prone of software early software testing were by developing a model with Levenberg-Marquardt (LM) algorithm based neural network tool for data collected from the PROMISE repository of empirical software engineering data then comparing the accuracy of LM with the polynomial function-based neural network. The experiment showed that Levenberg-Marquardt (LM) have higher accuracy (88.1%). So, that neural network based machine learning has good accuracy.

Saiqa Aleem et al.[24]**,** in this study they were used around fifteen data sets (AR1, AR6, CM1, KC1, KC3 etc) with various machine learning methods. Measured the performance of each method and finally conclude that SVM, MLP and bagging had high accuracy and performances.

According to Venkata U.B et al.[2] They evaluate different predictive model for real-time software defect data Sets. The experiment showed that there are not any exact techniques exist for each data set; however, IBL and 1 R were relatively better consistency in prediction accuracy compared to with other methods.

According to Martin Shepperd et al. [25] investigation, in order to predict and evaluate software defect, they were using a novel benchmark framework. In the evaluation stage, different learning schemes are evaluated according to that scheme selected. Then, in the prediction stage, the best learning scheme is used to build a predictor with all historical data and the predictor is finally used to predict defect in the new data.

VI. BASED ON CLUSTERING METHODS

Xi Tan et al.[26] experiments the software defect prediction model based on the function cluster for the purpose of improving the performance of the model. After applying this method, the researcher upgrades the performance 31.6 % to 99.2% recall and precision from 73.8 % to 91.6%.

Jaspreet Kaur et al.[27]investigated that, the fault proneness of object oriented programming via applying k-mean based clustering approach and finally they conclude as they have got 62.4% of accuracy.

Model building using clustering algorithms (EM and X-means) from three promise repository data (AR3, AR4, AR5) with an objective of predicting software faults. The first thing in experiment setting was normalizing the data set in to 0 to 1, then attribute selection algorithm applied that was CfsSubsetEval and without attribute reduction. Experiment result showed that X-means have accuracy (90.48) than other model for AR3 without attribute reduction[27] .

VII. BASED ON ENSEMBLE APPROACHES

Model building using ensemble approach were conducted by Shanthini et al., The purpose of this research was to address software fault prediction using ensemble approach. The data set was categorized in to three which are method level, class level and package levels. They were using NASA KC1 data for both method and class level metrics and eclipse data for package level with an ensemble methods (bagging, boosting, staking and voting). The experiment result shows that bagging perform better for method and package level data. Method level result using AUC- curve performance measurement was bagging (0.809), boosting (0.782), staking (0.79) and voting (0.63). Similarly, the performance measure of package level data using AUC-Curve was bagging (0.82), boosting (0.78), staking (0.72), and voting (0.76). In case of class level metric, the performance result could not similar to other metrics using AUC-Curve bagging (0.78), boosting (0.74), staking (0.8) and voting (0.82) [28].

According to Arvinder Kaur et al., the main purpose of the research was evaluating application of random forest for predicting fault prone class using open source software. The researcher used JEdit open source software with object oriented metrics to conduct studies. Based on the experiment result the accuracy RF is 74.24 % and its precision is 72 %, its recall is 79 %, its F-measure is 75 %, and its AUC is 0.81[29].

YI PENG et al., The goal of the paper was to assess the quality of ensemble approaches in software fault prediction with analytical hierarchal process. The researcher uses 13 different performance measures for 10 publicly NASA MDP data. An ensemble method used in this paper was Bagging, Boosting and Staking Based on the performance measure AdaBoost of decision tree gives best result accuracy 92.53 %, in this case decision tree is base classifier[30].

For more clarity, we summarized the above works with objective, methodology adopted, Contribution of studies and provide over all a remark for studies as Table IV below.

© www.ermt.net All Rights Reserved

Table IV: Summary of Related Works

Author (year)	**Objective of the study**	**Methodology/approaches/ tools/techniques used**	**Key findings**	**Remark**
Based on Classification Approaches				
EzgiErturk et al (2014)	- To predict software fault	- SVM, ANN and ANFS - 10-fold cross validation test mode - WEKA tool used	- Performance measure using SVM, ANN and ANFS were 0.7795, 0.8685, and 0.8573	- They used NASA MDP and maximum accuracy was 0.857 for data using ANFS method.
Malkit Singh et al (2013)	- Software defects prediction at an early Stage of the software development life cycle.	- Levenberg-Marquardt (LM) algorithm based ANN. And - Polynomial function (PF) based on ANN - MATLAB R2011a.	- Accuracy with LM based ANN was 88.1 % - Accuracy with PF based on ANN was 78.8 %	- They used Ant 1.7 data set and maximum accuracy they scored was 88.7 % using LM based ANN.
Saiqa Aleem et al. (2015)	- Comparative machine learning methods for publically available data using software prediction model.	- 10-cross validation test mode - MLP - SVM - Naïve Bayesian - AdaBoost - Bagging - Decision Tree - Random forest - KNN	- The mean accuracy for Software fault prediction model for the given 15 data - SVM (89.29 %) - Bagging (89.38 %) - Random forest (89.08 %)	- They used NASA MDP data set and the maximum accuracy was 99.52 for PC2 data using MLP method.
Venkata U.B et al. (2005)	- To predict software fault	- 70 % training and 30 % testing - Decision Tree - Naïve Bayes - Logistic Regression - Nearest Neighbor - 1-Rule - Neural Network - WEKA tool used - NASA MDP (CM1, JM1, KC1 and PC1)	- IBL and 1-R better consistency accuracy when compared with other methods - IBL ~ MAE (0.0543) - 1-R ~ MAE (0.0351)	- They used NASA MDP data and the maximum accuracy was 0.0543 using MAE performance on CM1 data
Author (year)	**Objective of the study**	**Methodology/approaches/tools/techniques used**	**Key findings**	**Remark**
Martin Shepperd et al. (2014)	- Factors for performance of software fault prediction accuracy	- Meta-analysis for the previous studies on software fault prediction such as all relevant and high quality primary studies.	- They conclude that defect prediction researches should conduct on blind analysis, improve reporting protocols and inter group protocols.	- It was survey on factor that affects the performance of software fault prediction accuracy.
Xi Tan et al. (2011)	- To improve the performance of software fault prediction model using recall and precision	- Eclipse 3.0 data - 90 to 10 % data split	- Cluster based defect prediction better than class based model in both recall and precision - Recall (31.6% to 99.2%) - Precision (73.8% to 91.6%)	- They used Eclipse 3.0 data and the maximum accuracy 99.2 % (Recall).
Qinbao Song et al. (2011)	- They proposed and evaluate a general frame work to evaluate software fault prediction model	- NASA MDP and AR Data sets - 90 to 10 % data split - Two data preprocessors - Two attribute selections - Naive Bayes - J48 - OneR	- The maximum accuracy scored on PC1 data set using Naïve Bayesian method (89.7%)	- They used NASA MDP and Art 1.7; the maximum accuracy was 89.7 for PC1 data using Naïve Bayesian.

© www.ermt.net All Rights Reserved

Based on Clustering Approaches				
Jaspreet Kaur et al. (2011)	- Predicting fault proneness of object oriented programming using K-mean	- KC1 data set used - WEKA - K-mean	- 62.4 % accuracy scored	- They used only KC1 data set and maximum accuracy was 62.4 using K-mean method.
Mikyeong Park et al. (2014)	- To predicting software faults	- Three promise repository data (AR3, AR4 and AR5) - EM - X-Means	- Xmeans have higher accuracy (90.48) for AR3 without attribute reduction.	- They used AR3, AR4 and AR5; the maximum accuracy was 90.48 for AR3 using Xmeans unsupervised.
Based on Ensemble Approaches				
Shanthini. A et al. (2013)	- software fault prediction using ensemble approach	- NASA KC1 data for both method and class level metrics - eclipse data for package level - They used WEKA - Bagging - Boosting - Staking - Voting	- Bagging perform better when compared with other approaches in case of method and package level - Bagging (AUC [0.809]) - Boosting (AUC [0.782]) - Staking (AUC [0.79]) - Voting (AUC[0.63]) - But not in class level metrics - Bagging (AUC[0.78]) - Boosting (AUC[0.74]) - Staking ([0.8]) - Voting ([0.82])	- They used KC1 method and class level data; the maximum accuracy was AUC-0.809 using bagging method.
Author (year)	**Objective of the study**	**Methodology/approaches/tools/techniques used**	**Key findings**	**Remark**
Arvinder Kaur et al. (2008)	- Evaluating application of random forest for predicting faults	- JEdit open source software data - OOP metrics	- Accuracy of RF was 74.24 % - Precision of RF was 72 % ,its recall was 79 % and its F-measure was 75 %	- They used JEdit open source and maximum accuracy was 74.24 % using RF method.
YI PENG et al. (2011)	- assess the quality of ensemble approaches in software fault prediction	- They used 13 different performance measure - 10 public NASA MDP - 10-fold cross validation - Bagging - Boosting - Staking - WEKA	- The maximum accuracy scored on AdaBoost (92 .53%)	- They used 10 NASA MDP data and maximum overall accuracy scored was 92.53 using AdaBoost

VIII. CONCLUSION AND FUTURE WORK

Now a day the development of software based system are increasing from the previous years due to its benefit. However, the quality of the system is required before it is delivered to end users. In order to enhance the software quality, we have various quality metrics such as software testing, CMM and ISO standards. Currently software testing becomes more and more important in the software reliability. Software defects prediction can effectively improve the efficiency of software testing and guide the allocation of resources. For the error-prone modules, we should spend more resource and time

The main objective of this study was to assess the previous research works with respect to software defect which applies machine learning method, data set used, tools they used, methodologies, their contribution to science and we classified it in to three such as based on classification, Clustering and ensemble methods. Finally, it is possible to extend this study by systematic literature review which includes books, dissertation, tutorial, Thesis.

REFERENCES

[1] Sandeep D and R. S, "Case Studies of Most Common and Severe Types of Software System Failure " *International Journal of Advanced Research in Computer Science and Software Engineering* vol. 2, pp. 341-347 August 2012

[2] Venkata U and R. A, "Empirical Assessment of Machine Learning based Software Defect Prediction Techniques " *Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems* 2005.

[3] Robert N, "Why Software Fails " 2005

© www.ermt.net All Rights Reserved

[4] Rajkumar G and K.Alagarsamy, "The Most Common Factors For The Failure Of Software Development Project," vol. 11, pp. 74-77, January 2013.
[5] L. J, "Major Causes of Software Project Failures " *CROSSTALK The Journal of Defense Software Engineering* pp. 9-12, July 1998.
[6] T. Clancy, "The Standish Group Report CHAOS," *Project Smart* pp. 1-16, 2014.
[7] Vikas S and J. R, "Cataloguing Most Severe Causes that lead Software Projects to Fail," *International Journal on Recent and Innovation Trends in Computing and Communication* vol. 2, pp. 1143– 1147, May 2014.
[8] Mikyeong P and E. H, "Software Fault Prediction Model using Clustering Algorithms Determining the Number of Clusters Automatically," *International Journal of Software Engineering and Its Applications*, vol. 8, pp. 199-204, 2014.
[9] Harry A, "machine learning capabilities, limitation and implications " *Technology Futures Researcher at Nesta* 2015.
[10] T. M. Mitchell, "Machine Learning " 1997.
[11] George T, Ioannis K, Ioannis P, and Ioannis V, "Modern Applications of Machine Learning " *Proceedings of the 1st Annual SEERC Doctoral Student Conference* vol. 1, 2006.
[12] Sarwesh S and S. K, "A Review of Ensemble Technique for Improving Majority Voting for Classifier," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 1, pp. 177-180, January 2013.
[13] Christian M, Gail K, and Marta A, "Title," unpublished|.
[14] D. Zhang, "Title," unpublished|.
[15] T. M. Mitchell, *The Discipline of Machine Learning*: Carnegie Mellon University, 2006.
[16] Xia C, Michael R, Kam-Fai W, and M. W, "ComPARE: A Generic Quality Assessment Environment for Component-Based Software Systems," *Center of Innovation and Technology the Chinese University of Hong Kong*, pp. 1-25.
[17] Ekbal R, Srikanta P, and V. B, "A Survey in the Area of Machine Learning and Its Application for Software Quality Prediction," *ACM SIGSOFT Software Engineering,*, vol. 37, September 2012.
[18] L. C. Briand, "Novel Applications of Machine Learning in Software Testing," pp. 3-10, 2008.
[19] Yogesh S, Pradeep K, and O. S, "A Review of Studies On Machine Learning Techniques," *International Journal of Computer Science and Security* vol. 1, pp. 70-84.
[20] Nguyen V, "The Application of Machine Learning Methods in Software Verification and Validation " MSc, Master of Science in Engineering, The University of Texas at Austin, Texas 2010.
[21] Ekbal R, Srikanta P, and V. B, "Software Quality Estimation using Machine Learning:Case-based Reasoning Technique " *International Journal of Computer Applications*, vol. 58, pp. 43-48, November 2012.
[22] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," *Expert Systems with Applications*, 2014.
[23] M. SURENDRA and D. N. GEETHANJALI, "Classification of Defects In Software Using Decision Tree Algorithm," vol. 5, pp. 1332-1340, June 2013.
[24] Saiqa A, Luiz F, and F. A, "Benchmarking Machine Learning Techniques for Software Defect Detection," *International Journal of Software Engineering & Applications*, vol. 6, pp. 11-23, May 2015.
[25] Martin S, David B, and T. H, "Researcher Bias: The Use of Machine Learning in Software Defect Prediction " *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, vol. 40, pp. 603-616, JUNE 2014.
[26] X. Tan, X. Peng, S. Pan, and W. Zhao, "Assessing Software Quality by Program Clustering and Defect Prediction," pp. 244-248, 2011.
[27] Jaspree K and P. S, "A k-means Based Approach for Prediction of Level of Severity of Faults in Software System," *Proceedings of International conference on Intelligent Computational Systems*, 2011.
[28] Pooja P and D. A. Phalke, "Software Defect Prediction for Quality Improvement Using Hybrid Approach," *International Journal of Application or Innovation in Engineering & Management*, vol. 4, June 2015.
[29] A. Kaur and R. Malhotra, "Application of Random Forest in Predicting Fault-Prone Classes," pp. 37-43, 2008.
[30] Y. I. Peng, G. Kou, G. Wang, W. Wu, and Y. Shi, "Ensemble of Software Defect Predictors: An Ahp-Based Evaluation Method," *International Journal of Information Technology & Decision Making*, vol. 10, pp. 187-206, 2011.

© www.ermt.net All Rights Reserved

View publication stats