International Conference on Intelligent Technologies & Science - 2021
(ICITS-2021)

A Review on Software Defect Prediction Using Machine Learning Techniques

Karishma Sahni[1], Dr. Ganesh Chandra[2]
Computer Science and Engineering,
Goel Institute of Technology & Management, Lucknow, India
karishmasahni903@gmail.com

Abstract: Software is playing an increasingly vital role in many industries. However, defects are not only inconvenient and annoying, but can also have serious consequences for software systems, especially for mission-critical systems. Therefore, software defect prediction models are useful for understanding, evaluating and improving the quality of a software system. Machine learning techniques have been employed to make predictions about the defectiveness of software components by exploiting historical data of software components and their defects. In order to predict software defects, many studies using distance-based classification algorithms with Mahalanobis distance function have been proposed. However, the common implementations of the Mahalanobis distance function do not take into account the label information (defective or defect-free) from the training data.

Keywords: ANSI, Machine Learning, Software Defect, SRGM

1. Introduction:

Business applications which are critical in nature require reliable software, but developing such software's is a key challenge which our software industry faces today. With the increasing complexity of the software these days, achieving software reliability is hard to achieve. The primary goal of software reliability modeling is to find out the probability of a system failing in given time interval or the expected time span between successive failures

Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment (ANSI definition). Software reliability modeling has gained a lot of importance in the recent years. Criticality of software in many of the present day applications has led to a tremendous increase in the amount of work being carried out in this area. The use of intelligent neural network and hybrid techniques in place of the traditional statistical techniques have shown a remarkable improvement in the prediction of software reliability in the recent years. Among the intelligent and the statistical techniques it is not easy to identify the best one since their performance varies with the change in data. SRGM has been used for predicting and estimating number of errors remaining in the software

ISO 9126 defines software quality as "the totality of features and characteristics of software product that bears on its ability to satisfy stated or implied needs"

While ISO 25000 takes the following approach to quality

"Capability of software products to satisfy stated and implied needs when used under specified conditions"

Software quality:- is the degree of conformance to explicit or implicit requirements and expectations.

- *Explicit*: clearly defined and documented
- *Implicit*: not clearly defined and documented but indirectly suggested
- *Requirements*: business/product/software requirements.
- *Expectations*: mainly end-user expectations.

Five of reasons quality is important to measure include:

- Safety – Poor quality in software can be hazardous to human life and safety.
- Cost – Quality issues cost money to fix.
- Customer Satisfaction (internal) – Poor quality leads stakeholders to look for someone else to do your job.
- Customer Satisfaction (external) – Software products that don't work, are hard to use.
- Future Value – Avoiding quality problems increases the amount of time available for the next project or the next set of features.

Based on these models, the Consortium for IT *Software Quality* has defined five major desirable structural characteristics *needed* for a piece of *software* to provide business value:

- Reliability,
- Efficiency,
- Security,
- Maintainability
- Size.

Software metrics it has been used for monitoring and controlling software process, asses and/or improves software quality, metrics collection and analysis is part of daily work activities in large software development organizations.

2. Related Work:

"Machine Learning Approach for Quality Assessment and Prediction in Large Software Organizations" (Rakesh Rana, and Miroslaw Staron) [1], *Importance of software is being rising day by day and its complexity such as* measuring, maintaining and increasing software quality. Software metrics

provide a quantitative means to measure and thus control various attributes of software systems. Assessing software quality early in the development process is essential to identify and allocate resources where they are needed most. Software quality, mainly the attributes related to dependability are even more important when developing software for systems deemed as safety, business and/or mission critical. Using machine learning techniques in conjunction to ISO/IEC 15939 measurement information model we can model overall quality of given software module/product/project. An algorithm or calculation combining one or more base and/or derived measures with associated decision criteria. It is based on an understanding of, or assumptions about, the expected relationship between the component measures and/or their behavior over time. Models produce estimates or evaluations relevant to defined information needs. The scale and measurement method affect the choice of analysis techniques or models used to produce indicators. As software becomes more integral part of our daily lives and its complexity increases - monitoring, assessing and improving software quality also becomes ever more important.

"Machine Learning-based Software Quality Prediction Models: State of the Art" (Hamdi A. Al-Jamimi and Moataz Ahmed) [2], *Quantification of parameters that affecting the quality of software and machine learning techniques being used to predict the quality of software. Software quality means a degree and satisfaction of the customers identified needs.*ML techniques have been utilized in many different problem domains as this field concentrates on building algorithms. Support Vector Machine, SVM for predicting the software fault-proneness modules. SVM being employed to the maintenance effort prediction. Bayesian network (BN) has been used in various studies in SWE area due its ability to integrate both empirical data and expert opinions. Neural networks (NN) as a tool for predicting the software faults. The relationship between the internal and external Quality attributes is surrounded with impression and uncertainty, different studies in the literature have made attempts to utilize the capability of fuzzy logic (FL) to estimate the software quality. current software quality prediction modelling approaches utilizing the two major sources of knowledge for building the models. Pre-requisite to effective combination of knowledge sources is the transparency of the model. The survey revealed that none of the current models address the model transparency issue.

"A Literal Review of Software Quality Assurance" (C. SenthilMurugan S. Prakasam) [3], Software development and maintenance is used to make the error-free Software and also concentrate on time-consuming and complex activity. To evaluate the quality of a software product and to keep its level high is much more difficult than to do them for the other industrial products. For maintaining the quality, performance, speed, efficiency and cost of the software the Software quality Assurance activities, principles and its methods are implemented in the early stages of software engineering development phases.A software system exists for one reason: to provide value to its users. Seven principles of software development process being performed to achieve the target. Different software applications require different approaches when it comes to testing, but some of the most common tasks in software QA include: PPQA audits, Peer Reviews, Validation testing, Data comparison, Stress testing, Conformance testing, Load testing, Usability testing, Robustness testing. the Software Quality Assurance concepts that are used to make the error-free Software and concentrate on complex activities and used to complete in time and in cost estimation is prevented. the early stages of software engineering development phases, because of this activity the software developer get the knowledge about the software what he is going to develop, it may reduce the rework and failures of the softwares and satisfy the customer's all requirements.

"Review of Improving Software Quality using Machine Learning Algorithms," (Jyoti Devi, Nancy Seghal) [4], Software is a process and maintains continuous change to improve the functionality and effectiveness of the software quality. During the life cycle of software various problems arises like advanced planning, well documentation and proper process control. This problem may result in not achieving the software quality as desired. With respect to competition in the market it is necessary to remove this problem with the help of software engineering. Quality software is reasonably bugs or a defect free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable. The software attributes are categorized two type's internal and external quality. The internal quality like efficiency, maintainability, testability, flexibility, reusability etc. The external quality like integrity, usability, reliability and accuracy etc. Software quality management split into three main activities: Quality assurance, Quality planning, quality control. A machine learning algorithms are developed to build machine learning models and important machine learning process. In this paper it has been discussed three classifier like decision tree, naïve bayes and support vector machine. Machine learning checks the prediction performance with the help of various performance measures: Precision, Recall, Accuracy, F-measure, Roc (receiver operating characteristics). In software development life cycle, maximum effort and cost is consumed on the testing and maintenance. the machine learning techniques like naïve bayes, decision tree and support vector machine algorithms been used for improving the software quality. Further different machine learning algorithm used to improve software quality and improve the performance in terms of precision, recall and Roc.
Several machine learning techniques such as ANNs, SVMs, CCNN, DTs, Group Method of Data Handling(GMDH) Polynomial network, Gene Expression Programming (GEP), Genetic Programming (GP), FIS and Dynamic evolving neuro-fuzzy inference system (DENFIS) have been proposed in the literature for solving various classification and

regression problems (Karunanithi et al. 1992; Kohavi 1995; Phillip 2003; Aggarwal et al. 2006, 2009; Jung Hua 2010; Ping and Hong 2006; Malhotra et al. 2009, 2011; Eduardo et al. 2010; Raj and Ravi 2008). There are few studies applied for the prediction of software reliability using machine learning methods based on past and present failure data of software. Some useful empirical studies based on multivariate linear regression and neural network methods have been carried out for prediction of software reliability growth trends. Although, multivariate linear regression method can address linear relationship but require large sample size and more independent variables (Jung Hua 2010). Many software reliability prediction models using ANNs have been applied for the prediction of software reliability successfully (Karunanithi et al. 1992; Singh and Kumar 2010c, d). However, effectiveness of neural network based prediction models depend on the behaviour of dataset that is basically of fluctuating nature. Therefore ANNs suffers from overfitting the results while dealing with real-life unknown data sets. Overfitting occurs usually when the parameters of a model are tuned in such a way that the model fits the training data well but it has poor accuracy when applied on separate data not used for training. The applications of SVM based machine learning approach in place of traditional statistical techniques has shown a remarkable improvement in the prediction of software reliability in the recent years (Xingguo and Yanhua 2007). SVM represents state of the art because of their generalization performance, ease of usability and rigorous theoretical foundations that practically can be used for modeling complex software failure behaviour. The design of SVM is based on the extraction of a subset of the training data that serves as support vectors and therefore represents a stable characteristic of the data. The major limitation of SVMs is the increasing computational and storage requirement with respect to the number of training examples (Chen et al. 2008). Ping and Hong (2006) investigated the capabilities of SVMs for the prediction of software reliability with the help of simulated annealing algorithms (SA). In their study it is suggested that SVM model with simulated annealing algorithms (SVMSA) results in better predictions than other existing techniques in practice. Yang and Xiang (2007) suggested an SVM-based model for software reliability prediction and pointed out that failure data collected from early phases of software development life cycle is more appropriate to be used which affect prediction accuracy. Xingguo and Yanhua (2007) investigated the status of early prediction methods for software reliability by introducing SVM. They identified that early prediction model based on SVM is more accurate in its prediction with better capability of generalization. The main advantage of DTs are their descriptive nature, which allows practitioners to interpret the model's decision easily compared to other machine learning techniques such as ANNs and SVMs. While DTs do show their strengths in various real-life applications, these methods have rarely been used for predicting software reliability in practice. The capability of fuzzy logic systems leads to the achievement of more efficient and decisive system in software reliability prediction. Due to large computation and low learning rate of prediction model, the machine learning techniques using FIS is found to be more effective than classical machine learning (Mueller and Lemke 1999). However, the present challenge is to make it even more efficient by incorporating a fairly new technique which can improve the prediction rate and require less computational resources.

SRGM has been used for predicting and estimating number of errors remaining in the software. The primary goal of software reliability modeling istofind outtheprobability ofa system failingin given time interval or the expected time span between successive failures. In our study, we attempt to empirically assess the use of ANFIS for predicting the software failures. Other ML techniques used for predicting software reliability are FFBPNN, GRNN, SVM, MLP, Bagging, CFBPNN, IBK, Lin Reg, M5P, RepTree, M5Rules. Although ANN, SVM
etc. has been previously used in literature (Xingguo and Yanhua 2007) but for the first time ANFIS has been applied to cumulative week failure dataset. In this work, in order to make the most realistic and efficient comparison we have also analyzed the same data sets for above mentioned ML techniques. The background of using ANFIS was that if it had proven empirically to predict the software failures with least errors in comparison to other above mentioned techniques, then it may possibly be used as a sound alternative to other mentioned existing techniques for software reliability predictions. Also other above mentioned ML techniques were empirically analysed for the first time together on five different types of data sets taken altogether.

Several ML techniques have been proposed and applied in the literature for software reliability modelling and forecasting. Some of the techniques are Genetic Programming, Gene Expression Programming, Artificial Neural Network, Decision Trees, Support Vector Machines, Feed Forward Neural Network, fuzzy models, Generalized Neural Network etc. (Malhotra et al. 2011; Xingguo and Yanhua 2007; Hua Jung 2010; Karunanithi et al. 1992; Singh and Kumar 2010d; Eduardo et al. 2010; Cai et al. 1991; Specht 1991). Karunanithi et al. (1992) carried out analysis of detailed study to explain the use of connectionist models in the reliability growth prediction for the software's. Cai et al. (1991) focused on the development of fuzzy software reliability models instead of probabilistic software reliability models as he says that reliability is fuzzy in nature. Ho et al. (2003) carried out a comprehensive study of connectionist models and their applicability to software reliability prediction and inferred that these are better as compared to traditional modes. Su and Huang (2006) had applied neural network for predicting software reliability. Madsen et al. (2006) focused on the application of Soft Computing techniques for software reliability prediction. Pai and Hong (2006) performed experiments using SVMs for forecasting software reliability. Despite of recent advances in this field, it was observed that

different models have varied predictive reliability capabilities. Lou et al. (2009) discusses about the software reliability prediction using relevance vector machine. Yang et al. (2010) develops a hybrid model using model mining techniques and genetic algorithms for software reliability prediction. Lo (2011) discusses about the utilization and applicability of Auto-Regressive Integrated Moving Average technique and SVM for reliability prediction. Kumar and Singh (2012) discusses about the usage of machine learning techniques like cascade correlation neural network, decision trees and fuzzy inference system to predict the reliability of software products. Torrado et al. (2013) described the usage of Bayesian model together with Gaussian processes to estimate and predict the number of software failures over time. Park et al. (2014) talks about the applicability of data driven methods to identify an appropriate multi-step prediction strategy for software reliability. Liu et al. (2015) discusses the applicability of hybridization of Singular Spectrum Analysis method and Auto Regressive Integrated Moving Average methodology for prediction of medium and long-term software failures. Lou et al. (2016) carried out study to estimate future occurrences of software failures to aid in maintenance and replacement using Relevance vector machines which are kernel-based learning methods.

There are many studies about software bug prediction using machine learning techniques. For example, the study in [5] proposed a linear Auto-Regression (AR) approach to predict the faulty modules. The study predicts the software future faults depending on the historical data of the software accumulated faults. The study also evaluated and compared the AR model and with the Known power model (POWM) used Root Mean Square Error (RMSE) measure. In addition to, the study used three datasets for evaluation and the results were promising. The studies in [6], [7] analyzed the applicability of various ML methods for fault prediction. Sharma and Chandra [3] added to their study the most important previous researches about each ML techniques and the current trends in software bug prediction using machine learning. This study can be used as ground or step to prepare for future work in software bug prediction. R. Malhotra in [8] presented a good systematic review for software bug prediction techniques, which using Machine Learning (ML). The paper included a review of all the studies between the period of 1991 and 2013, analyzed the ML techniques for software bug prediction models, and assessed their performance, compared between ML and statistic techniques, compared between different ML techniques and summarized the strength and the weakness of the ML techniques. In [9], the paper provided a benchmark to allow for common and useful comparison between different bug prediction approaches. The study presented a comprehensive comparison between a well-known bug prediction approaches, also introduced new approach and evaluated its performance by building a good comparison with other approaches using the presented benchmark.

D. L. Gupta and K. Saxena [10] developed a model for object-oriented Software Bug Prediction System (SBPS). The study combined similar types of defect datasets which are available at Promise Software Engineering Repository. The study evaluated the proposed model by using the performance measure (accuracy). Finally, the study results showed that the average proposed model accuracy is 76.27%. Rosli et al. [11] presented an application using the genetic algorithm for fault proneness prediction. The application obtains its values, such as the object-oriented metrics and count metrics values from an open source software project. The genetic algorithm uses the application's values as inputs to generate rules which employed to categorize the software modules to defective and non-defective modules. Finally, visualize the outputs using genetic algorithm applet. The study in [12] assessed various object-oriented metrics by used machine learning techniques (decision tree and neural networks) and statistical techniques (logical and linear regression). The results of the study showed that the Coupling Between Object (CBO) metric is the best metric to predict the bugs in the class and the Line Of Code (LOC) is fairly well, but the Depth of Inheritance Tree (DIT) and Number Of Children (NOC) are untrusted metrics. Singh and Chug [13] discussed five popular ML algorithms used for software defect prediction i.e. Artificial Neural Networks (ANNs), Particle Swarm Optimization (PSO), Decision Tree (DT), Naïve Bayes (NB) and Linear Classifiers (LC). The study presented important results including that the ANN has lowest error rate followed by DT, but the linear classifier is better than other algorithms in term of defect prediction accuracy, the most popular methods used in software defect prediction are: DT, BL, ANN, SVM, RBL and EA, and the common metrics used in software defect prediction studies are: Line Of Code (LOC) metrics, object oriented metrics such as cohesion, coupling and inheritance, also other metrics called hybrid metrics which used both object oriented and procedural metrics, furthermore the results showed that most software defect prediction studied used NASA dataset and PROMISE dataset. Moreover, the studies in [14], [15] discussed various ML techniques and provided the ML capabilities in software defect prediction. The studies assisted the developer to use useful software metrics and suitable data mining technique in order to enhance the software quality. The study in [16] determined the most effective metrics which are useful in defect prediction such as Response for class (ROC), Line of code (LOC) and Lack Of Coding Quality (LOCQ). Bavisi et al. [17] presented the most popular data mining technique (k-Nearest Neighbors, Naïve Bayes, C-4.5 and Decision trees). The study analyzed and compared four algorithms and discussed the advantages and disadvantages of each algorithm. The results of the study showed that there were different factors affecting the accuracy of each technique; such as the nature of the problem, the used dataset and its performance matrix. The researches in [18], [19] presented the relationship between object-oriented metrics and fault-proneness of a class.

Singh et al. [20] showed that CBO, WMC, LOC, and RFC are effective in predicting defects, while Malhotra and Singh [21] showed that the AUC is effective metric and can be used to predict the faulty modules in early phases of software development and to improve the accuracy of ML techniques. This paper discusses three well-known machine learning techniques DT, NB and ANNs. The paper also evaluates the ML classifiers using various performance measurements (i.e. accuracy, precision, recall, F-measure and ROC curve). Three public datasets are used to evaluate the three ML classifiers. On the other hand, most of the mentioned related works discussed more ML techniques and different datasets. Some of the previous studies mainly focused on the metrics that make the SBP as efficient as possible, while other previous studies proposed different methods to predict software bugs instead of ML techniques.

In the last few years many research studies has been carried out in this area of software reliability modeling and forecasting. They included the application of neural networks, fuzzy logic models; Genetic algorithms (GA) based neural networks, recurrent neural networks, Bayesian neural networks, and support vector machine (SVM) based techniques, to name a few. Cai et al. (1991) advocated the development of fuzzy software reliability models in place of probabilistic software reliability models (PSRMs). Their argument was based on the proof that software reliability is fuzzy in nature. A demonstration of how to develop a fuzzy model to characterize software reliability was also presented. Karunanithi et al. (1992) carried out a detailed study to explain the use of connectionist models in software reliability growth prediction. It was shown through empirical results that the connectionist models adapt well across different datasets and exhibit better predictive accuracy than the well-known analytical software reliability growth models. Sitte (1999) made a comparative study of neural networks and parametric-recalibration models in software reliability prediction and found neural networks to be much simpler to use and also to be better predictors. Also, through empirical results it was shown that the neural network models are better trend predictors. Ho et al. (2003) performed a comprehensive study of connectionist models and their applicability to software reliability prediction and found them to be better and more flexible than the traditional models. A comparative study was performed between their proposed modified Elman recurrent neural network, with the more popular feedforward neural network, the Jordan recurrent model, and some traditional software reliability growth models. Numerical results show that the proposed network architecture performed better than the other models in terms of predictions. Despite of the recent advancements in the software reliability growth models, it was observed that different models have different predictive capabilities and also no single model is suitable under all circumstances. Tian and Noore (2005a) proposed an on-line adaptive software reliability prediction model using evolutionary connectionist approach based on multiple-delayed-input single-output architecture. The proposed approach, as shown by their results, had a better performance with respect to next-step predictability compared to existing neural network model for failure time prediction. Tian and Noore (2005b) proposed an evolutionary neural network modeling approach for software cumulative failure time prediction. Their results were found to be better than the existing neural network models. It was also shown that the neural network architecture has a great impact on the performance of the network. According to Bai et al. (2005) Bayesian networks show a strong ability to adapt in problems involving complex variant factors. They developed a software prediction model based on Markov Bayesian networks, and a method to solve the network model was proposed. Reformat (2005) proposed an approach leading to a multi technique knowledge extraction and development of a comprehensive meta-model prediction system in the area of corrective maintenance of software. The system was based on evidence theory and a number of fuzzy-based models. In addition they carried out a detailed case study for estimating the number of defects in a medical imaging system using the proposed approach. Pai and Hong (2006) have applied support vector machines (SVMs) for forecasting software reliability where simulated annealing (SA) algorithm was used to select the parameters
of the SVM model. The experimental results show that the proposed model gave better predictions than the other compared methods. Su and Huang (2006) showed how to apply neural networks to predict software reliability. Further they made use of the neural network approach to build a dynamic weighted combinational model (DWCM) and experimental results show that the proposed model gave significantly better predictions. Also recently, neural networks were applied for predicting faults in object-oriented software (Kanmani et al., 2007). The study showed neural network models to be performing much better than the statistical methods. Application of intelligent techniques in place of the statistical techniques has increased by leaps and bounds in the recent years. Application of Soft Computing techniques in software reliability engineering has come up recently (Madsen et al., 2006). Despite the recent advancements in the software reliability growth models, it was observed that different models have different predictive capabilities and also no single model is suitable under all circumstances. An ensemble uses the output obtained from the individual constituents as inputs to it and the data is processed according to the design of the arbitrators.

3. Conclusion:

We are confident that the research makes positive contributions towards the goal of improving the performance of software defect prediction models. Nevertheless, there are still practical issues that should be addressed to increase the adoption of our approach in practice. In reality, apart from class imbalance, data sets extracted from software archives often contain much correlated information, and pair with random errors and noise. These lead to a problem called over-

fitting occurring when a defect prediction model becomes excessively complex.

References:

[1]. Rakesh Rana, and Miroslaw Staron (Issue 2015), "Machine Learning Approach for Quality Assessment and Prediction in Large Software Organizations," IEEE Transaction on https://ieeexplore.ieee.org/document/7339243.

[2]. Hamdi A. Al-Jamimi and Moataz Ahmed(Issue 2013),"Machine Learning-based Software Quality Prediction Models: State of the Art," IEEE Transaction on https://ieeexplore.ieee.org/document/6579473/

[3]. C. SenthilMurugan S. Prakasam "A Literal Review of Software Quality Assurance,"
International Journal of Computer Applications (0975 – 8887) Volume 78 – No.8, September 2013

[4]. Jyoti Devi, Nancy Seghal "Review of Improving Software Quality using Machine Learning Algorithms," IJCSMC, Vol. 6, Issue. 3 March 2017.

[5]. A. Sheta and D. Rine, "Modeling Incremental Faults of Software Testing Process Using AR Models ", the Proceeding of 4th International Multi-Conferences on Computer Science and Information Technology (CSIT 2006), Amman, Jordan. Vol. 3. 2006.

[6]. D. Sharma and P. Chandra, "Software Fault Prediction Using MachineLearning Techniques," Smart Computing and Informatics. Springer, Singapore, 2018. 541-549.

[7]. R. Malhotra, "Comparative analysis of statistical and machine learning methods for predicting faulty modules," Applied Soft Computing 21, (2014): 286-297 [5] Malhotra, Ruchika. "A systematic review of machine learning techniques for software fault prediction." Applied Soft Computing 27 (2015): 504-518.

[8]. D'Ambros, Marco, Michele Lanza, and Romain Robbes. "An extensive comparison of bug prediction approaches." Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on. IEEE, 2010.

[9]. Gupta, Dharmendra Lal, and Kavita Saxena. "Software bug prediction using object-oriented metrics." Sādhanā (2017): 1-15..

[10]. M. M. Rosli, N. H. I. Teo, N. S. M. Yusop and N. S. Moham, "The Design of a Software Fault Prone Application Using Evolutionary Algorithm," IEEE Conference on Open Systems, 2011.

[11]. T. Gyimothy, R. Ferenc and I. Siket, "Empirical Validation of ObjectOriented Metrics on Open Source Software for Fault Prediction," IEEE Transactions On Software Engineering, 2005.

[12]. Singh, Praman Deep, and Anuradha Chug. "Software defect prediction analysis using machine learning algorithms." 7th International Conference on Cloud Computing, Data Science & EngineeringConfluence, IEEE, 2017.

[13]. M. C. Prasad, L. Florence and A. Arya, "A Study on Software Metrics based Software Defect Prediction using Data Mining and Machine Learning Techniques," International Journal of Database Theory and Application, pp. 179-190, 2015.

[14]. Okutan, Ahmet, and Olcay Taner Yıldız. "Software defect prediction using Bayesian networks." Empirical Software Engineering 19.1 (2014): 154-181.

[15]. Bavisi, Shrey, Jash Mehta, and Lynette Lopes. "A Comparative Study of Different Data Mining Algorithms." International Journal of Current Engineering and Technology 4.5 (2014).

[16]. Y. Singh, A. Kaur and R. Malhotra, "Empirical validation of objectoriented metrics for predicting fault proneness models," Software Qual J, p. 3–35, 2010.

[17]. Malhotra, Ruchika, and Yogesh Singh. "On the applicability of machine learning techniques for object oriented software fault prediction." Software Engineering: An International Journal 1.1 (2011): 24-37.

[18]. A.TosunMisirli, A. se Ba¸ S.Bener,"A Mapping Study on Bayesian Networks for Software Quality Prediction", Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, (2014).

[19]. T. Angel Thankachan1, K. Raimond2, "A Survey on Classification and Rule Extraction Techniques for Data mining",IOSR Journal of Computer Engineering ,vol. 8, no. 5,(2013), pp. 75-78.

[20]. T. Minohara and Y. Tohma, "Parameter estimation of hyper-geometric distribution software reliability growth model by genetic algorithms", in Proceedings of the 6th International Symposium on Software Reliability Engineering, pp. 324–329, 1995.