CHAPTER 2

REVIEW OF DATA MINING TECHNIQUES FOR SOFTWARE DEFECTS PREDICTION

In this chapter the various proposals made in the literature for software defects prediction is studied. The literature study carried out in this chapter can be broadly classified into:

- Data mining techniques for defect prediction.
- Techniques to improve software reliability based on metrics.
- Neural network based defect prediction.

2.1 Data mining techniques for Defect Prediction

Yi Liu et al., 2010 [2] investigated the problem of software quality classification modeling using the history of metric dataset obtained from single software project. The classification modeling got from a single dataset is generally not adequate to build strong and an accurate model. To tackle the issue, software quality classification modeling was done using multiple datasets sourced from different software projects. Previous study has demonstrated that using multiple datasets for validation can achieve robust genetic programming-based models. The effectiveness of the modeling using multiple datasets is extensively studied in this paper. Moreover, a
novel general purpose based classifier consisting of training, multiple-dataset validation, and voting phases, is proposed. The datasets used for experimentation were obtained from NASA software projects. The performance by the proposed classifier was compared with the results of seventeen other data mining techniques. The comparison shows that the proposed approach is more effective and accurate with the use of multiple datasets.

Song et al., 2006 [18] proposed prediction of defect associations and defect correction effort based on association rule mining methods. Test resources are more effectively allocated for detecting software defects. The proposed method was applied to more than 200 projects. The experiment results show the accuracy achieved is high for both defect association prediction and defect correction effort predictions. The result of the proposed method was also compared with PART, C4.5 AND Naïve Bayes methods. The comparison shows that the proposed method accuracy was higher by at least 23 percent.

Lessmann et al., 2008 [10] investigated the performance of classification algorithm. To compare the software defect prediction, experiments were conducted using 10 public domain datasets from NASA Metric Data repository, using 22 classifiers. The general impression is that the predictive accuracy metric based classification is useful. The results also indicated that the importance attached to particular classification algorithms was not significant as generally
assumed. The results showed that there was no significant difference among the top 17 classifiers.

Munson et al., 1992 [43] investigated statistical based methods like discriminating analysis for the detection of fault prone programs. In this paper it was proposed to implement principal components to reduce multicollinear complexity metrics to uncorrelated measures on orthogonal complexity domains. The transformed data was used to classify the programs. Eleven software metrics were computed from the programs and data prepared for the classification engine. The misclassification rate was 10 percent showing a high degree of classification.

Tom M. Mitchell 1997, [38] attempted to classify data using Naïve Bayesian algorithm. Naïve Bayes is one of the popularly used learning algorithms in data mining and machine learning. It is popular because of its efficient and effective inductive learning algorithms. Classification based on Naïve Bayes algorithms gives very competitive performance due to its conditional independence assumption.

Ohlsson et al., [41] derived metrics using design documents of telecommunication software modules to predict fault prone modules prior to testing with accurate results.

Menzies et al., 2004 [29] proposed Naïve Bayes learners for studying the defect detectors from static code measures. Comparison
of Naïve Bayes learners and entropy-based decision tree learner was done to show the effectiveness of Naïve Bayes learners. The study concludes that accuracy was not an effective way to assess those detectors. When using Naïve Bayes learners on heavily stratified data, 200-300 examples are enough to learn adequate detectors.

Riquelme et al., 2009 [5] used the promise repository to obtain the software metrics program dataset and proposed a genetic algorithm search for rules characterizing subgroups with a high probability of being defective. The genetic algorithm handles the problem of unbalanced datasets efficiently especially when the unbalanced sets consists of more non-defective samples than the defective samples.

Catal et al., 2007 [12] modeled Artificial Immune System based on the Human immune system for defect prediction. The proposed classifier imitates the behavior of the antigen and the antibody during an attack by pathogens into the human biological system. The evolution of the immune system to new attacks is modeled to solve the software defect prediction problem.

Drown et al., 2009 [4] proposed Fenton et al., 1999 [36] proposes a holistic approach to select classifiers for software defect prediction. The study highlights serious flaws in current approaches to software defects prediction. The issues affecting defect prediction can be summarized as follows:
• The unknown correlation linking defects and failures.
• Problems with the “multivariate” statistical approach.
• Problems of using size and complexity metrics as sole “predictors” of defects.
• Problems in statistical methodology and data quality.
• False claims about software decomposition and
• The “Goldilock’s Conjecture”.

Evolutionary Sampling, a genetic algorithm based data sampling method, to improve software quality modeling for high-assurance systems. The proposed sampling was compared with the existing data sampling techniques. Two case studies of software quality model both before and after applying the Evolutionary sampling technique was presented. The improvement in the performance of software quality models by using Evolutionary sampling is shown empirically.

2.2 Techniques to predict defects and reliability based on metrics

Achcar et al., 1991 [45] approach to the software reliability prediction extended the Bayesian approach by using Poisson distribution to propose a novel software reliability model. The Bayesian inference models used Metropolis-within-Gibbs algorithms
for the Moranda’s model. Model selection was based on the predictive density.

Xin Jin et al., 2006 [22] attempted to provide software reliability for Software Engineering Management and made a list of metrics and implemented them in a common dataset. This research work tries to improve all the measures made by them by incorporating additional metrics with a combined effort. Good results were got from experimenting with the artificial immune recognition system of classifiers.

Andersson et al., 2007 [11] proposed a novel method for analysis of fault distributions in software systems. The proposed method used replicated quantitative analysis which was mathematically modeled and proved in detection of fault distributed across complex software system.

Basili et al., 1996 [39] extensively investigated design metrics for object oriented software with the understanding that errors multiply at each stage of the software design process. The idea is to identify metrics at the design stage so that prediction can be done earlier to remove defects. The proposed method was validated with good results.
Emam et al., 2001 [32] based on the previous work on predicting faulty modules using object oriented design metrics proposed enhancement in the computation of design metrics. The proposed method of feature extraction performed better than the previous models; however the disadvantages of the proposed method were also highlighted.

Selby 1990 [46] successfully implemented software metrics for predicting software module reuse. The study analyzed four classes of software modules:

- Modules reused without revision.
- Modules reused with slight revision (< 25% revision).
- Modules reused with major revision (> 25% revision).
- Newly developed modules.

Comparison of development variable in various modules was done through nonparametric statistical models. Factors contributing for the successful reuse of software are the module implementation factors and the module design factors. Evaluation of the fault rate of the reused, modified, and newly developed modules using the design and implementation factors was summarized.

Cartwright et al., 2000 [33] investigated an industrial object-oriented (OO) system made of 133,000 lines of C++ using empirical methods. The data system was a subsystem of a telecommunication
The study showed that the OO constructs such as inheritance, polymorphism are not really useful. The study found that the classes in inheritance structures were three times more defect prone than the classes that are not in inheritance structures. Prediction systems were constructed using a number of states and events per class. Though the prediction systems have only local significance, the need of suites of metrics in OO technology is not required; thus measurement technology becomes more accessible.

Kim et al., 2008 [9] proposed change classification method for predicting dormant software bugs. Change classification was based on machine learning classifiers, which helps in determining the similarity of change to previous buggy changes or clean changes. The presences of bugs were predicted using the change classification. The classifier is trained using the features from revision history of the software, these when applied classifies the changes in the software as buggy or clean. The results showed 78% accuracy. The change classification was superior as it had small prediction granularity and semantic information of source code was not required for classification. Change classification works on a broad array of programming languages.

Liu et al., 2010 [2] proposed a novel genetic programming based on search approach for software quality modeling with multiple software project repositories. The training on multiple projects provides a cross project perspective on software quality modeling which can effectively sum up the quality trends of the development
organization. This approach includes three strategies using Baseline classifier, Validation classifier and Validation-and-Voting classifier. Case study of software metrics and defect data using 7 systems showed that the Validation-and-Voting classifiers are better software quality model. The paper also presents another case study consisting of 17 different machine learners using majority-voting approach for predicting fault proneness class of program modules. This study provides a clear direction for organization’s software measurement data repositories for improving software quality modeling

2.3 Neural network based defect prediction

Ebru Ardl et al., 2009 [3] investigated the modules of feed forward neural network. The feed forward neural networks were the first and the simplest type of artificial neural network. Faults are mainly found in the modules of the neural network, the study investigates the most severely affected modules in comparison with other modules.

Jianhong, et al (2010) explored five Neural Network Based techniques and comparative analysis is performed for the modeling of severity of faults present in function based software systems. The NASA's public domain defect dataset is used for the modeling. The comparison of different algorithms is made on the basis of Mean Absolute Error, Root Mean Square Error and Accuracy Values. It is
concluded that out of the five neural network based techniques Resilient Backpropagation algorithm based Neural Network is the best for modeling of the software components into different level of severity of the faults. Hence, the proposed algorithm can be used to identify modules that have major faults and require immediate attention.

Yuan, et al (2011) presented a method to evaluate the software reliability using Fuzzy-Neural network. In order to improve the accuracy of the evaluation, this paper established a reliability prediction model based on adaptive-network based fuzzy inference system (ANFIS). The model use the reliability data(defect counts of every thousand code lines) of one software project as input data, and use the prediction of reliability as output data, training Adaptive-Fuzzy neural network, get the membership function of defect counts of every thousand code lines.

Bezerra, et al (2007) introduced a novel algorithm based on constructive RBF neural networks aimed at predicting the probability of errors in fault-prone modules; it is called RBF-DDA with Probabilistic Outputs and is an extension of RBF-DDA neural networks. The advantage of the proposed method is that it informs the test team of the probability of defect in a module, instead of indicating only if the module is fault-prone or not. Experiments carried out with static code measures from well-known software defect datasets from NASA show the effectiveness of the proposed method. The performance of the proposed method for software defect prediction is
compared with kNN and two of its variants, the S-POC-NN and R-POC-NN. The experimental results showed that the proposed method outperforms both S-POC-NN and R-POC-NN and that it is equivalent to kNN in terms of performance with the advantage of producing less complex classifiers.

2.4 Conclusion

In this chapter various techniques for selection of software metrics and defect classification proposed in literature was studied. Proposed methods included classification and clustering techniques. Naïve Bayes algorithm has been extensively used for classification by many researchers in the literature. Extensive work has also been done in the areas of feed forward neural network. Though many of the articles investigated emphasized on the metrics, not many emphasized on classification algorithms specifically for defect prediction of software modules. Further investigation need to be carried with emphasis on preprocessing of data and classifiers specifically designed for defect prediction in software modules. The said need as driven us to seek the solution for the problem.