An Efficient Software Fault Prediction Scheme to Assure Qualified Software Implementation using Improved Classification Methods

Rajkumar N, Viji C

Abstract: ***Software quality is a main concern of software developers to ensure the required software that can provide user required services. However quality of software would be degraded considerably due to presence of the software faults in the programming language. Detection and removal of software faults requires more concern to be taken which needs to be concentrated more for improved performance. In the existing research work, accurate software fault prediction is done by introducing two stage data pre-processing stage which would select the more important features from the training data set and will result with the optimal training dataset thus the training accuracy can be improved. However existing research method doesn't concentrate the dependencies between software modules and it doesn't focus on the classification performance. These challenges are highlighted in the newly introduced research methodologies to obtain the accurate software prediction outcome by introducing the novel proposed research methodology namely Optimal and eliable Prediction of Software Faults (ORPSF).***

In the proposed research methodology, Optimal feature selection is performed by considering the inter relationship between the different features using Genetic Algorithm. This technique would select the optimal features which can detect the software faults accurately than the existing research methods. And the SVM classification approach is introduced to perform classification which can learn the training instances more accurately. Thus software fault prediction can be done accurately. The overall implementation of the proposed research technique is performed in the java simulation environment from which it can be shown that the proposed research methodology yields optimal results compared to the available research techniques.

Keywords: Software fault prediction, feature selection, Training software instances, optimal analysis, Software quality

I. INTRODUCTION

Quality of the software is a primary factor for any software company. Software defect prediction is essentially a data mining process that helps in quality improvement [1]. Software fault prediction is considered to be a core process in software engineering used for improving the quality and guarantee of software in minimal amount of time and reduced expenditure [2]. Its implementation is done prior to the testing stage of the software development life cycle. Also, software defect prediction models help in delivering the defects or the number of defects.

Software defect prediction has inspired lot of researchers to yield various models with a project or cross project to boost the different quality and monitoring guarantee of software. Two schemes are available to develop a software defect prediction model such as supervised learning and unsupervised learning [3].

Supervised learning faces the challenge of training the software defect prediction model, which requires past data or few outcomes that are known. The training performance of the model present within the project is good, however it results in challenges in knowing about other novel projects. Several openly available datasets that freely are accessible for the researcher such as PROMISE, Eclipse and Apache exists to get over the challenge while the training is carried out on a fresh project. Several researchers have focused on building a cross project defect prediction model having several metrics set such as class level metric, process metrics, static code metrics, however they may not be able to develop more practically accurate models [4].

Several classifiers or learning algorithm are used for selecting a huge array of software metrics such as Naïve Bias, Support Vector Machine, Random Tree, J48 and Logistic Regression. These classifiers have reached various resourceful conclusions. Nearly, all of the current software prediction models have been developed with the help of non-simple metrics using which the prediction model attained the accuracy with satisfaction. In this research work, the contribution made is associated with the present position of research. It also demonstrated the prediction model having the simple group of metrics used for feature selection. It also suggested that the software prediction model developed with minimal set of metrics can attain the desired outcome.

The general software defect prediction process adopts the machine learning technique, [5][6][7]. The first step in the prediction process is to get the instances from the software, and an instance could be code, function, class or method etc. These instances could be generated from various issue tracking system, version control system or e-mail records. An instance uses various metrics that is obtained from the software. These instances could be classified in buggy B or the number of bugs and clean C or number of clean ones.

Once the instances are recognized with the group and metrics, the first step of machine learning pre-processing approaches made use on instances for creating a new similar kind of instance. The pre-processing is used for the extraction of the features, scaling of the data and eliminating the noise [8][9][10]. However, it is not mandatory to use it on every kind of defect prediction model [11].

Revised Manuscript Received on May 22, 2019.

Rajkumar N, SVS College of Engineering, Coimbatore, Tamilnadu, India

Viji.C, SVS College of Engineering, Coimbatore, Tamilnadu, India

Once the pre-processing of the instances are over, the new instances that are generated for training the defect prediction model. The prediction model renders the outcome in terms of buggy instances and clean instances. The number of bugs present in an instance is called as regression. It generates just two results for the instances, either buggy or clean and therefore it is also called to be binary classification.

In the proposed research methodology, Optimal feature selection is performed by considering the inter relationship between the different features using Genetic Algorithm. This technique would select the optimal features which can detect the software faults accurately than the existing research methods. And the SVM classification approach is introduced to perform classification which can learn the training instances more accurately. Thus software fault prediction can be done accurately.

The overall organization of the research approach is given as below: Section 2 provides discussion on the various associated research works that has been conducted. In section 3, the newly introduced research technique is discussed in elaborate manner with the appropriate examples and explanation. Section 4 provides the experimental analysis of the novel research technique. At last, in section 5, overall research works is concluded in terms of suitable performance metrics.

II. RELATED WORKS

Bibi et al. [12] used Regression via Classification (RvC) for the issue of software fault prediction in order to get the estimation of the number of software defects along with a confidence interval. RvC rendered much superior regression error count compared to the standard regression techniques. Pekka dataset obtained from a big commercial bank in Finland and ISBSG (International Software Benchmarking Standards Group) dataset were utilized for the evaluation. The performance evaluation metric included Mean Absolute Error (MAE).

Bingbing, Qian, Shengyong, and Ping [13] employed Affinity Propagation clustering algorithm over two datasets and their performance comparison is done with that of K-means clustering technique. Affinity Propagation was designed by Brendan J. Frey and Delbert Dueck. This algorithm outperformed K-means clustering over two datasets as per the Type-II error. Also, the performance evaluation metrics included Type-I error, Type-II error, and the whole correct classification rate (CCR).

Catal and Diri[14] highlighted on the high-performance defect predictors that depends on machine learning like Random Forests and algorithms, which again is dependent on a novel computational intelligence scheme known as Artificial Immune Systems. Openly available NASA datasets were brought into use. It was stated that Random Forests yields superior prediction performance for huge datasets and Naive Bayes is considered to be the best among prediction algorithms for smaller datasets in terms of the Area under Receiver Operating Characteristics Curve (AUC) evaluation parameter.

Turhan and Bener [15] revealed that independence assumption made in Naive Bayes algorithm does not yield degrading effect with principal component analysis (PCA) pre-processing and PD, PF and balance parameters were used in their research work.

Chang, Chu, and Yeh[16] introduced a fault prediction technique that depends on association rules for the discovery of fault patterns. It was proven that the prediction results were commendable. The advantage incurred of this technique is that the fault patterns discovered could be utilized in causal analysis to know about the reasons behind the defects.

Tosun, Turhan, and Bener[17] carried out experiments on openly available datasets for the validation of Zimmermann and Nagappan's article issued in ICSE'08. Three embedded software projects were exploited for the evaluation purpose. It was mentioned that network measures are important indicators of fault-affected modules for huge systems. And the performance evaluation metrics included PD, PF, and precision.

Turhan, Kocak, and Bener[18] examined 25 projects of a telecommunication system and then the models are trained on NASA MDP data. Static call graph based ranking (CBGR) and nearest neighbor sampling is used for building the fault predictors. It was found that nearly 70% of faults could be found by examining just 6% of code with Naive Bayes model and 3% of code with CBGR model.

III. OPTIMAL AND RELIABLE SOFTWARE FAULT PREDICTION SYSTEM

Software fault prediction has a critical role to play in the software engineering framework which is required by the software developers for the implementation of an efficient software. Analysis and prediction of software faults would be a more tedious task that is highlighted in this technical work. Classification is one among the popular techniques utilized for software defect prediction. Its primary task involves the categorization of modules, denoted by a set of software metrics, into two classes that include fault-prone (FP), or non-fault-prone (NFP). For a certain classification model, the classifier has to be trained in prior on the basis of the training data acquired through the mining performed of past software archives, like change logs in software configuration management, bug reports in bug tracking systems, and the developers' e-mails.

In the proposed research work, optimal and reliable prediction of software faults is done by introducing the novel framework namely Optimal and Reliable Prediction of Software Faults (ORPSF).In the proposed research methodology, Optimal feature selection is performed by considering the inter relationship between the different features using Genetic Algorithm. This technique would select the optimal features which can detect the software faults accurately than the existing research methods. And the SVM classification approach is introduced to perform classification which can learn the training instances more accurately. Thus software fault prediction can be done accurately. The elaborate description of the newly introduced research technique is illustrated in the section that follows:

Dataset Description

For evaluating the efficacy of the novel scheme, a set of experiments is designed and performed on datasets gathered from real-world software projects, inclusive of the Eclipse projects,

and the NASA software projects that are generally employed by researchers in software defect prediction. The Eclipse datasets are acquired from the PROMISE data warehouse, whereas the NASA datasets are acquired from the openly accessible MDP (Metrics Data Program) database.

In the case of the Eclipse datasets, three releases of Eclipse (i.e., Eclipse2.0, Eclipse 2.1, and Eclipse 3.0) are gathered with instances that are measured at the Java class level. Each one of the Eclipse datasets has 198 features, inclusive of the code complexity measures (like LOC, cycloramic complexity, and number of classes),the syntax tree based measures, and several others.

Opitmal Feature Selection Using Genetic Algorithm

The first step involves carrying out the relevance analysis to eliminate the unnecessary features, and the second step involves conducting redundancy management to remove repetitive features. Feature ranking is employed for the first step, in which every feature is ranked based on the importance(i.e., weights) in distinguishing the instances of various classes (i.e., FP or NFP). Then the subset of features out of the top- of the ranking list is chosen.

Feature selection refers to a process in which a subset of the real features are chosen. The efficacy of any system can be measured with the help of its classification accuracy. With the expansion in the dimensionality of a domain, there is also an increase in the number of features. Getting an optimal subset of features is revealed to be NP-hard [3]. A problem is termed NP hard when resolving it in polynomial time would render the possibility of solving all the problems in class NP in polynomial time. Considering any type of diagnosis process on not every real feature can always be advantageous for classification or regression tasks. There can be as many features, which don't have any relevance or are noisy in the dataset distribution. These features have the capability of reducing the classification performance. The feature selection procedure is required for the classification or regression issues for achieving a superior accuracy with minimized subset of features. If the features are decreased, it also leads to an automatic reduction of the cost, which is necessary for performing the test.

Genetic algorithms are one among the best means for solving a set of problems for which very less information is provided. Genetic algorithms are quite generic algorithms and therefore will function well in any kind of search space. All that is required to be known is what is required for the solution to be capable to perform well, and a genetic algorithm will be capable of creating a superior solution. Genetic algorithms make use of the principles of selection and evolution to generate solution for different sophisticated problems. Genetic algorithms tends to work well in an environment in which there exists a massive set of candidate solutions and in which the search space is not desirable and has several ups and downs.

As a matter of fact, genetic algorithms can perform well in any kind of environment, but they might be usually outperformed by more condition oriented algorithms in the much simple search spaces. Hence, it has to be remembered that genetic algorithms are not among the best of the preferences at all times in random situations. At times, they might take some time to execute and are therefore not always a practical approach for real time usage. However, they are one among the most potential techniques with which high quality solutions are generated quickly for a problem. There are few basic methodology/Terminology that will be used while implementing genetic algorithm like:

Individual – Probable solutions

Population - Set of all of the individuals

Search Space - All probable solutions for the specified problem

Chromosome – Blueprint for an individual

Trait - Probable aspect of an individual entity

Allele - Probable settings for a trait

Locus - The position of a gene on the chromosome

Genome - Set of all the chromosomes for an individual entity

Feature optimization is the primary topic in this research. In this research, Genetic algorithm (GA) is helpful in the optimization of the features from dataset. According to PengfeiGuo, Xuezhi Wang, and Yingshi Han [12] genetic algorithm (GA) is a strong random algorithm, the initial idea of GA is the application of natural selection and natural genetics in machine learning and optimization problems. For resolving a problem, GA uses a population known as strings or chromosomes and GA transforms the population using few genetic operators to seek a close to optimal solution to the problem. An example of genetic operators is selection, crossover, and mutation.

Selection: According to previous research a selection approach is applied on every individual to decide the way in which individual are selected for mating on the basis of their fitness value. Fitness, by itself can be defined to be the capability of an individual to sustain and reproduce in the environment. After selection

is applied, it creates a new population from the old one, thereby beginning a new generation. Every chromosome is assessed in this generation to decide on its fitness value. This fitness value is then utilized to determine which chromosomes to be used from the population for creating the next generation.

Crossover: Once selection is over, crossover operation is used for the chosen chromosomes. Crossover comprises of swapping the genes or sequences between two individuals. Crossover operation is then iterated with diverse parent individuals until the next subsequent generation has sufficient individuals.

Mutation: After the crossover process, mutation operator is used on some randomly chosen subset of the population. Mutation modifies the chromosomes and introduces new traits. Mutation is used for bringing versatility in the population.

According to Engelbrect [14] fitness function is the capability of an individual of an evolutionary algorithm to survive. To calculate fitness function. The formula to calculate fitness function can be seen below.

$$w = error(M)$$
$$FV = w/(N - n1)$$

w is the classification error , M is a classification model produced by machine learning classification algorithm, FV is the Fitness Value, N refers to the number of features present in the data, and n1 stands for the number of features selected that is employed in the classification. The process step to find the fitness value in general can be described below:

1. A population consist of one row and all features with 0 and 1 value is passed to the fitness function
2. Find (FI) which is the features has value 1 in the population
3. Get X1 which is all row of the data used but only with the features in F1
4. Calculate n1 which is the number of features selected in F1
5. Calculate M by classify X1 using selected classification algorithm
6. Calculate the classification error (w)
7. Calculate the fitness value (FV)

c. Prediction Of Software Faults Using Support Vector Machine

SVMs are helpful methods used for data classification. As with SVM, every object is defined by a vector Xi consisting of N real numbers (features or descriptors) that are associated with a point present in a multidimensional space. Each object present in the first class (positive) are assigned with a value of Y_i= +1, and those in the second class are assigned with Y_i=-1. In the linearly separable scenarios, the objects can be rightly categorized by

$$w.x_i + b \geq +1 \text{ for } y_i = +1 \text{(class1)}$$
$$w.x_i + b \leq 1 \text{for } y_i = -1 \text{(class2)}$$

Where w refers to a vector normal to the hyper plane and b stands for a scalar value. The SVM tries to get an optimal separating hyper plane with maximum margin resolving the optimization problem below:

$$\max_{w,b} \frac{2}{\|w\|} \text{ subject to } y_i(w.x_i + b) - 1 \geq 0$$

The concepts above can also be improved to linearly non separable cases, in which no hyper- plane can be utilized to exactly isolate the sets of points. In this scenario, non-negative slack variables i=1, 2, 3….m. can be introduced so that

$$w.x_i + b \geq +1 - \xi_1 \text{ for } yi = +1$$
$$wx_i + b \leq -1 + \xi_i \text{ for } y_i = -1$$

The objective here is to get a hyper plane, which renders the minimum number of training errors i.e. to reduce the violation of constraint. The equation that needs to be solved is:

$$\max_{w,b} \frac{2}{\|w\|} + c\sum_{i=1}^{m} \xi_i \text{ subject to } y_i(w.x_i + b) - 1 + \xi_i \geq 0$$

Where c refers to a user predefined penalty parameter. The parameter c exhibits a significant influence on the SVM classifier's accuracy, and therefore has to be selected with care. The non-linear(non-) separable cases could just be modified to linear cases through the projection of the input variable onto new dimensional feature space making use of a kernel function K (X_i,X_j). Various kernel functions inclusive of polynomial, radial basis function (RBF) and sigmoid kernel have been proposed. But, radial basis function is found to be the most extensively employed kernel function and it demonstrates superior performance in many cases. Therefore, we also have utilized the Radial basis kernel function. This technique can help in efficiently predicting the software defects in the effective manner in terms of different data set features.

IV. EXPERIMENTAL RESULTS

This section provides the numerical assessment of the newly introduced research technique in terms of different performance measures for analyzing the improvement in performance of the newly introduced and the available research strategies. The java simulation environment is utilized in the implementation of the newly introduced research scheme. The performance measures taken into consideration in this research article are grouped as: "Accuracy, Sensitivity, Specificity, and Precision". The implementation of the novel research technique is done in the mat lab simulation environment that is assessed for its performance in terms of the performance metrics. The comparison analysis is carried out between the novel technique, Optimal and Reliable Prediction of Software Faults (ORPSF).and the available techniques KNN, ANN.

Accuracy: It is defined to be the degree of right detection of drug names existing in the tweets. That indicates less false positive rate. The accuracy of the newly introduced system has to be greater compared to the other available techniques like IDM and SVM. The accuracy value is computed in terms of the true positive, false positive, true negative and false negative values of the drug prediction system. The accuracy is computed as given below:

$$\text{Accuracy} = \frac{T_p}{(T_p + F_p + F_n)}$$

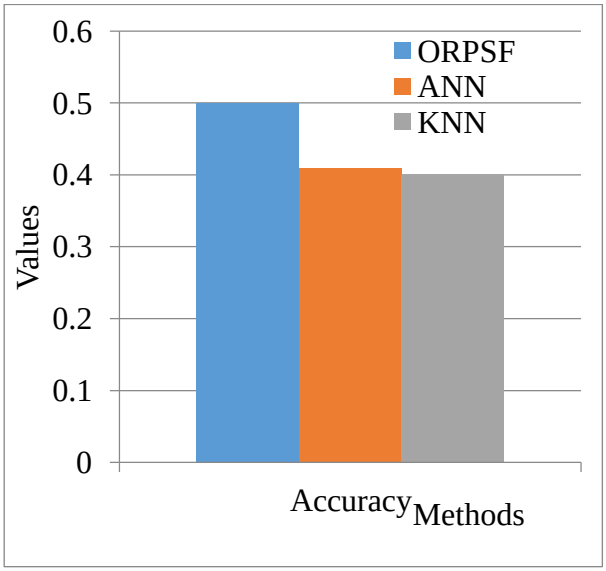

Fig. 1 Accuracy comparison

Figure 1 shows the graphical representation of the comparison analysis of the accuracy metric. This graphs reveals that the newly introduced research approach is better in comparison with the available research techniques. ORPSF is 22% better compared to ANN, which is 25% more than KNN.

Sensitivity: It is also referred as true positive rate that is specified to be the degree of right classification of the drug data items to be positive. The instance of a medical test to be helpful in identifying a disease, and the sensitivity of the test is defined as the ratio of individuals who test positive for the

disease amongst those persons who are affected by the disease. In mathematical terms, this can be computed as:

$$\text{Sensitivity} = \frac{\text{number of true positives}}{\text{Number of true positives+Number of false negatives}}$$

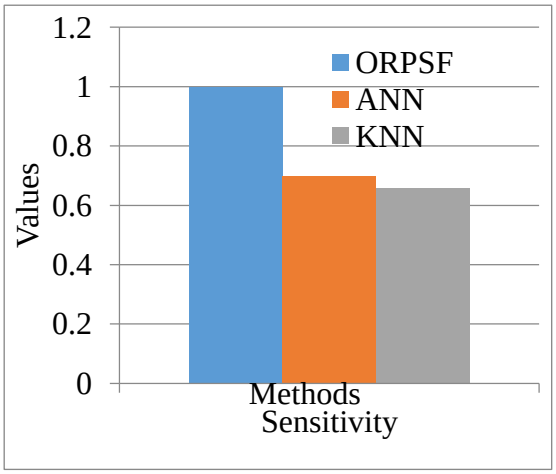

Fig. 2 Sensitivity comparison

Figure 2 provides the graphical representation of the sensitivity metric comparison analysis. This graph reveals that the proposed research technique is better compared to the available research methodologies. ORPSF is 43% more than ANN, 52% higher than KNN.

Specificity: It is also referred as true negative rate that is defined to be the degree of rightly categorizing then on drug data items. Take the instance of a medical test for the diagnosis of a disease. Specificity of a test is the ratio of healthy patients who are known not to be affected by the disease, and who will test negative for it. In mathematical terms, this can also be expressed as:

$$\text{Specificity} = \frac{\text{number of true negatives}}{\text{Number of true negatives+Number of false positives}}$$

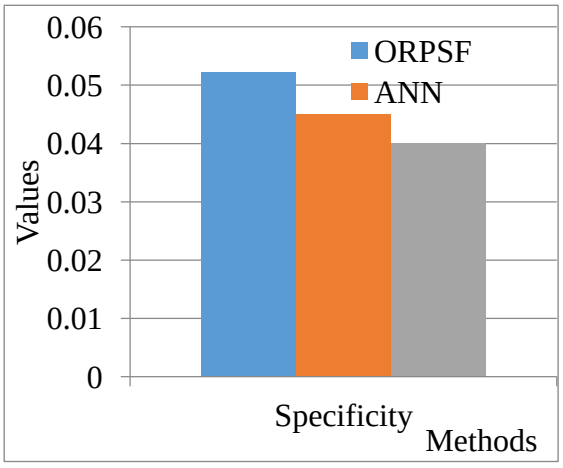

Fig. 3 Specificity comparison

Figure 3 shows the graphical representation of the specificity metric comparison analysis. This graph reveals that the newly introduced research approach is better compared to the other available research scheme. OSPRF is 16% better compared to ANN, 31% better compared to KNN.

Precision: It is the ratio of the instances retrieved, which are relevant

$$\text{Precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

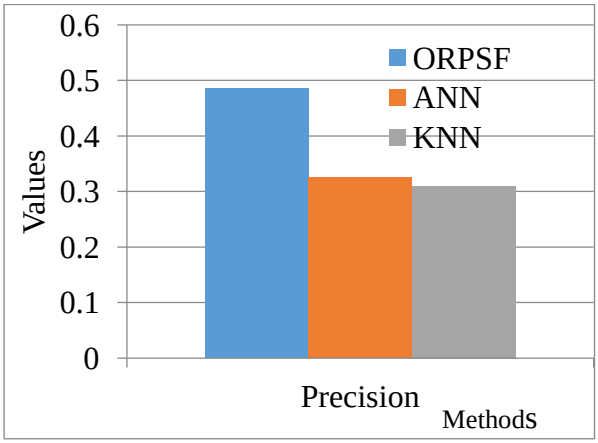

Fig. 4 Precision comparison

Figure 4 graphically represents the comparison analysis of the precision metric. This graph reveals that the newly introduced research approach is better compared to the other available research schemes. ORPSF is 50% better compared to ANN, 57% better compared to KNN.

V. CONCLUSION

Software fault prediction is termed the most required task in the software testing phase which attempts to provide the secured and reliable environment for the software developers to implement and output the software to the users. This is assured in the newly introduced research methodology by presenting the framework called as Optimal and Reliable Prediction of Software Faults (ORPSF) which attempts predict the software faults accurately. In the proposed research methodology, Optimal feature selection is performed by considering the inter relationship between the different features using Genetic Algorithm. This technique would select the optimal features which can detect the software faults accurately than the existing research methods. And the SVM classification approach is introduced to perform classification which can learn the training instances more accurately. Thus software fault prediction can be done with accuracy. The overall implementation of the novel research methodology is carried out in the java simulation environment from which it can be shown that the novel research methodology helps in providing optimal results compared to the other available research methodologies.

REFERENCES

1. Steidl, D., Deissenboeck, F., Poehlmann, M., Heinke, R., &Uhink-Mergenthaler, B. (2014, September). Continuous software quality control in practice. In Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on (pp. 561-564). IEEE.
2. Miguel, J. P., Mauricio, D., & Rodríguez, G. (2014). A review of software quality models for the evaluation of software products. arXiv preprint arXiv:1412.2977.
3. Laradji, I. H., Alshayeb, M., &Ghouti, L. (2015). Software defect prediction using ensemble learning on selected features. *Information and Software Technology*, *58*, 388-402.
4. Kumar, R., & Gupta, D. L. (2015). Software Fault Prediction: A Review.

5. Bacchelli, M. D'Ambros, and M. Lanza. Are popular classes more defect prone? In Proceedings of the 13th International Conference on Fundamental Approaches to Software Engineering, FASE'10, pages 59– 73, Berlin, Heidelberg, 2010. Springer-Verlag
6. E. Engstrom, P. Runeson, and G. Wikstrand. An empirical evaluation of regression testing based on fixcache ¨ recommendations. In Software Testing, Verification and Validation (ICST), 2010 Third International Conference on, pages 75–78, April 2010.
7. N. Nagappan and T. Ball. Use of relative code churn measures to predict system defect density. In Proceedings of the 27th international conference on Software engineering, ICSE '05, pages 284–292, 2005
8. S. Kim, H. Zhang, R. Wu, and L. Gong. Dealing with noise in defect prediction. In Proceeding of the 33rd international conference on Software engineering, ICSE '11, pages 481–490, New York, NY, USA, 2011. ACM
9. T. Menzies, J. Greenwald, and A. Frank. Data mining static code attributes to learn defect predictors. IEEE Trans. Softw. Eng., 33:2–13, January 2007.
10. J. Nam, S. J. Pan, and S. Kim. Transfer defect learning. In Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, pages 382–391, Piscataway, NJ, USA, 2013. IEEE Press.
11. T. Jiang, L. Tan, and S. Kim. Personalized defect prediction. In Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on, pages 279–289, Nov 2013.
12. Bibi, S., Tsoumakas, G., Stamelos, I., &Vlahvas, I. (2008). Regression via classification applied on software defect estimation. Expert Systems with Applications, 34(3), 2091–2101
13. Bingbing, Y., Qian, Y., Shengyong, X., & Ping, G. (2008). Software quality prediction using affinity propagation algorithm. In IJCNN 2008 (pp. 1891–1896).
14. Catal, C., &Diri, B. (2009a). Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. Information Sciences, 179(8), 1040–1058
15. Turhan, B., &Bener, A. (2009). Analysis of Naive Bayes' assumptions on software fault data: An empirical study. Data Knowledge Engineering, 68(2), 278–290
16. Chang, C., Chu, C., &Yeh, Y. (2009). Integrating in-process software defect prediction with association mining to discover defect pattern. Information and Software Technology, 51(2), 375–384.
17. Tosun, A., Turhan, B., &Bener, A. (2009). Validation of network measures as indicators of defective modules in software systems. In Proceedings of the 5th international conference on predictor models in software engineering (Vancouver, British Columbia, Canada, May 18–19, 2009). PROMISE '09 (pp. 1–9). New York, NY: ACM.
18. Turhan, B., Kocak, G., &Bener, A. (2009). Data mining source code for locating software bugs: A case study in telecommunication industry. Expert Systems and Application, 36(6), 9986–1900.