Prediction of Software Defects Based on Software Metrics Using Support Vector Machine

R. Sathya[1], Dr. P. Sudhakar[2]
[1, 2] Department of Computer Science & Engineering
[1, 2] Annamalai University, Annamalai Nagar, India

Abstract- *The prediction of defects in the software plays a key role in enhancing the software quality and eventually it helps to reduce the cost and time for software testing. The defect prediction in software is considered as an essential skill required during the software project planning. The defect prediction is a complex task and it needs great effort. The prediction is carried out using the available dataset containing software metrics collected from the defective and non-defective software modules. Conventionally the software metrics are used for defining the program complexity and estimating the time required to develop the software. At present to increase the software-quality extensive researches are being carried out for the prediction of defects in a software module. This research work will help software developers to find defects on the developed code using the available software metrics. This in turn helps them to identify modules that require further attention. This kind of predictions will help to increase the reliability of the software. The objective of this research work is to increase the prediction accuracy by selecting an optimal subset of attributes using the Genetic Algorithm. The feature The Support Vector Machine (SVM), k- Nearest Neighbor (kNN), and Gaussian Mixture Model (GMM) are the classifiers used in this work.*

Keywords- Defect Prediction, software metrics, software quality, Genetic Algorithm, support vector machine.

I. INTRODUCTION

At present the software systems are highly complex and versatile and hence it requires great effort to develop and test it. Therefore it is very essential to find and eliminate software defects. Accurate prediction of defects in software modules using the software metrics will help to enhance the software quality. Software metrics are the measure of property of a part of the software and they are measure during the design and development phase of the software project. Software metrics are used for the evaluation of the quality of the software. The defect in a software product can be due to mismatch between the requirements mentioned by the end user and the developed code. To put it in other way, a defect in software product is a bug in the code or the program logic which makes the software to malfunction or to generate inappropriate results. The prediction of defects in the software is the task of identifying defects in the software modules. To produce high quality software without any defects this early prediction of defects will help to a greater extent and also this will lead to reduction in the development cost. The rework effort required when defects are detected after testing will be reduced. Hence to achieve the software quality this defect prediction is essential. The software metrics used to predict defects can be categorized in to two types namely code and process metrics. These metrics are used to build defect prediction model. The development team can use these models to predict defects in the software modules during the initial phases of the software development. Software metrics are usually utilized to evaluate the potential of the software to meet the required objective. Code metrics namely size, McCabe [1], Halstead [2], Object Oriented (OO), and CK are used frequently than the process metrics. The Cyclomatic Complexity metric measures the structural complexity of the code whose value depends on the number of unique code path in the developed software. The halsteads metrics denotes the quantitative measures of the code complexity with respect to the volume of the operands, operators, and length of the program [3]. The product metrics has the lines of code (LOC) which indicates the approximate number of line in the code. OO metrics help to find faults, and helps developer to make their classes and objects simpler [4].

Many researchers have proposed various techniques to detect or predict defects in the developed code using static code metrics. These techniques include the conventional statistical methods such as logistic regression and the machine learning algorithms like ANN, Bayesian Classifiers, Decision trees, SVM [5]. In [6] a novel method using Adaptive Neuron Fuzzy Inference System (ANFIS) is proposed to predict software defects. They achieved a classification accuracy of 77.95, 86.85, and 85.73 when classifiers such as support vector, neural network and ANFIS respectively are trained using McCabe metrics. In [7] using Ward neural network and General Regression neural network (GRNN), prediction of number of defects in a class and prediction of number of lines changed per class are carried out respectively. In [8] the authors focused on the classification analysis rather than the classification performance. The basic objective is to find how the training data are being classified by the Support Vector Machine. In [9] Iterative Dichotomiser algorithm was used for

the classification of the defective modules from non-defective modules based on metrics namely Volume, Program length, Difficulty, Effort and Time Estimator. In [10] addressed two practical issues of a software defect prediction process. It is generally difficult to collect a huge volume of data for training a model and there are much less defective modules than defect less modules. Therefore, the training data will be an imbalanced dataset, so they proposed a semi supervised training algorithm called Random Committee with Under Sampling (ROCUS)

II. DATASET AND SOFTWARE METRICS

The dataset used in this study for experiments are from NASA software projects and they are publicly available for research purpose under NASA IV&V Facility Metrics Data Program. The dataset JM1 [11] is from software projects written in a procedural language (C)where the software is divided in to modules called functions. The dataset comprises10885 samples with 21 software metrics (independent variables) representing a module of the software and a Boolean variable to mention whether the attributes represent a defective or non-defective module. The 21 independent attributes includes McCabe metrics, Halstead metrics, Line Count, and Branch Count. Table 1 lists these metrics.

Table 1. List of Software Metrics

Metric	Type	Definition
V(g)	McCabe	Cyclomatic Complexity
EV(g)	McCabe	Essential Complexity
IV(g)	McCabe	Design Complexity
LOC	McCabe	Total lines of Code
N	Derived Halstead	Total Number of operands and operators
V	Derived Halstead	Volume on minimal Implementation
L	Derived Halstead	Program Length = V/N
D	Derived Halstead	Difficulty = 1/L
I	Derived Halstead	Intelligent Count
E	Derived Halstead	Effort to write program = V/L
B	Derived Halstead	Effort Estimate
T	Derived Halstead	Time to write program = E/18 s
LOCode	Line Count	Number of lines of statement
LOComment	Line Count	Number of lines of comment
LOBlank	Line Count	Number of lines of blank
LOCodeAndComment	Line Count	Number of lines of code and Comment
UniqOp	Basic Halstead	Number of Unique operators
UniqOpnd	Basic Halstead	Number of Unique operands
TotalOp	Basic Halstead	Total number of operators
TotalOpnd	Basic Halstead	Total number of operands
BranchCount	Branch	Total number of branch count

III. FEATURE SELECTION USING GENETIC ALGORITHM

To overcome the problem of high dimensional feature space and to increase the classification accuracy by removing the redundant and irrelevant features two methods are generally used feature extraction and feature selection. The main objective of both feature extraction and feature selection is to reduce the dimension of the feature vector and thereby reducing the time required to train the model. The feature selection deals with the task of selecting the best feature set which reduces the classifier training time and as well as increase the classification accuracy. From a given a set of *d* features, the feature selection algorithm selects a subset of size *m* which increases the classification accuracy. If the size of the feature set is *d* then there will be 2^d possible feature subsets. The selection of best feature subset can be viewed as a combinatorial optimization problem and is solved using Genetic Algorithms [12]. The GA's are insensitive to noise and they are considered to be an excellent choice for the basis of a more robust feature selection strategy for improving the performance of the classification system. For the successful implementation of any searching problem using GA requires an suitable representation and a sufficient function to evaluate. For the problem of feature selection, the importance of selecting the appropriate representation and evaluation function is essential. The simple scheme for representation is the binary representation by which, all the features of the candidate feature set is treated as a binary gene. Also each feature consists of a fixed-length string of constant length in binary form which represents certain subset of the given feature set.

An initial set of features provided to the GA serves as input and as a training set which represents positive and negative training samples of the defective and non-defective modules. A search procedure is used to explore the space of all subsets in the given feature set. The performance of all the selected feature subsets is calculated by using the evaluation function and the classification result is also calculated. The optimal feature subset found is then given as output as the best set of features that can be used to train the classifiers of the prediction system. The overall fitness function is evaluated using the equation given below,

$$F = \sum_{i=1}^{N} S_i * W_i - \sum_{j=n+1}^{M} S_j * W_j \qquad (1)$$

IV. CLASSIFIERS FOR SOFTWARE DEFECT PREDICTION

The three classifiers used in this work are Support Vector Machine, k-Nearest Neighbors, and Gaussian Mixture Model. All three classifiers don't have built-in feature selection ability and are commonly used in the machine learning applications. Support Vector Machine (SVM) is a mathematical model which has a supervised learning algorithm capable of analyzing data and identifying patterns in it. The classification process is done in two steps one is training phase and the other one is testing phase. The labeled data are fed as input during the training time and the data to be classified is given in the testing phase. The accuracy of classification depends on the efficiency of the trained model. In this work the support vector machine with Gaussian kernel is used and it is denoted by the equation,

$$K(x,y) = \exp\left(\frac{-\|x-y\|^2}{\sigma^2}\right) \quad (2)$$

From the results obtained it is being inferred that the SVM classifier performed best when compared to other classifiers. The *k*-Nearest Neighbors algorithm (or k-NN for short) is a non parametric method that can be used to classify *N*-dimensional data. The input for the algorithm is a*k* closest training examples in the feature space and the algorithm produces class membership as the result. An unknown feature vector of data is classified based on the class of its neighbors i.e. the feature vector is assigned to the class which is more common among its *k*-nearest-neighbors. The value of *k* positive and typically it is small. If *k* = 1, then the vector is simply assigned to the class of that single nearest neighbor. A commonly used distance metric and the metric used in this work is Euclidean distance.

The Gaussian Mixture Model (GMM) is a useful supervised learning classification algorithm which can be used for classification of *N*-dimensional dataset. During the training phase GMM is built for each class of data. In this work two GMM has to be constructed to fit the defective and non-defective data samples. There will not be any interactions between GMM of different classes. At the classification phase the unknown-class data is given as input to GMM of each class. The predicted class is the one associated with the GMM with the maximum probability.

V. EXPERIMENTS AND RESULTS

The block diagram of the proposed methodology for software defect prediction is presented in the Fig. 1. The feature selection is used to find a subset of the highly discriminant features. It selects features that are capable of discriminating samples that belong to defective and non-defective classes. In this software defect prediction problem, the classifier will be given a subset of the dataset of known class on which training is run, and a dataset of unknown class against which the model is tested. The cross validation is used to partition the dataset into complementary subsets, performing the analysis on one subset called the training set, and validating the analysis on the other subset called the testing set. The training dataset is given as input to the classifier along with the class label to obtain a trained model. Finally the test set is fed to the trained model and results are compared against the expected output. The performance measures used to analyze the efficiency of the classifiers are Classification Accuracy, Precision, Recall, and F-Measure.

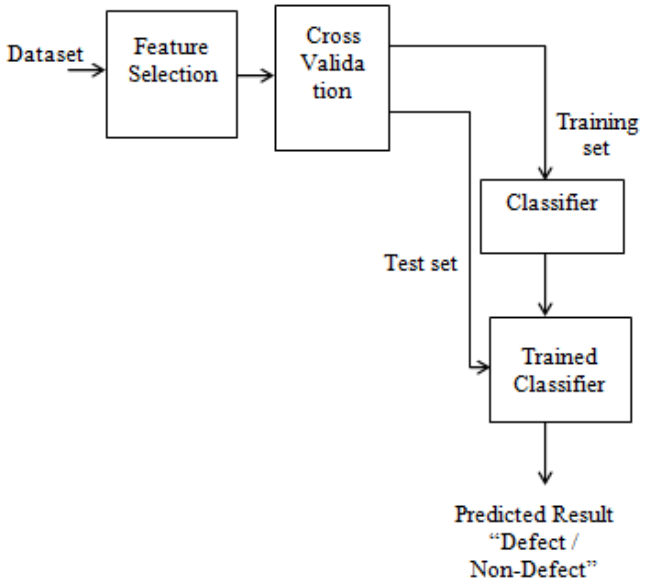

Fig. 1. Block diagram of the defect prediction process

Precision

Precision measures the correctness of the prediction or classification process. Precision is the ratio of the number of samples correctly predicted as defective to the total number of samples predicted as defective. It is calculated as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3)$$

Recall

Recall can be called as defect detection rate and it is the ratio of the number of samples correctly predicted as defective to the total number of samples that are actually defective. It can be mathematically represented as below

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4)$$

F-measure

F-measure considers both precision and recall equally important by taking their harmonic mean It is calculated as follows

$$F-Measure = \frac{2*Precision*Recall}{Precision+Recall}$$

In the experiments with PROMISE datasets, the objective is to compare the classifiers constructed using feature subsets selected by the genetic algorithm with those that use the entire set of attributes available. The Fig.2 shows a sample plot of fitness value for each generation when only 18 attributes are selected from the whole feature set.

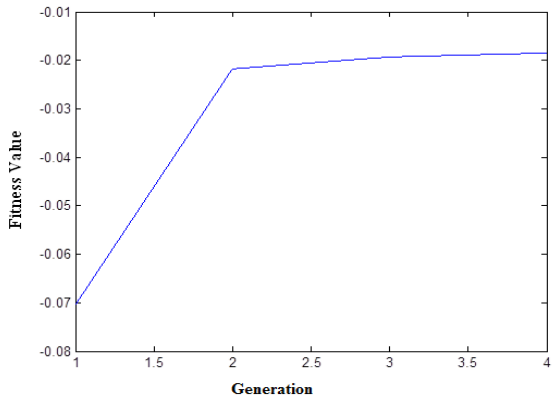

Fig. 2 Plot of Fitness value for each generation

Table 2. Prediction Result for JM Dataset with 21 attributes

Classifier	Accuracy	Precision	Recall	F-Measure
SVM	87.54	88.48	98.00	0.929
KNN	81.25	89.28	88.12	0.886
GMM	84.67	90.21	90.36	0.902

Table 3. Prediction Result for JM Dataset with selected 18 attributes

Classifier	Accuracy	Precision	Recall	F-Measure
SVM	90.69	90.66	100.0	0.951
KNN	83.27	91.36	90.02	0.906
GMM	86.74	92.49	92.90	0.926

Table 4. Prediction Result for JM Dataset with selected 15 attributes

Classifier	Accuracy	Precision	Recall	F-Measure
SVM	88.96	89.36	99.12	0.939
KNN	82.63	90.48	89.18	0.898
GMM	85.72	91.52	91.78	0.916

The feature selection presents superior performance and performs significantly better than feature set with no feature selection. The classification models built with 18 features outperformed models built with fewer features or more features. Among all the classifiers used SVM exhibits better performance. The Fig.3 presents the comparison of the prediction accuracy of different classifiers when using different number of attributes. When the number of attribute is reduced from 21 to 18 there is much progress in the prediction accuracy but when it is again reduced to 15 the accuracy starts decreasing. This shows that the set of 18 attributes is an optimal subset of attributes which helps to construct a better prediction model for efficiently predicting defects in the software modules.

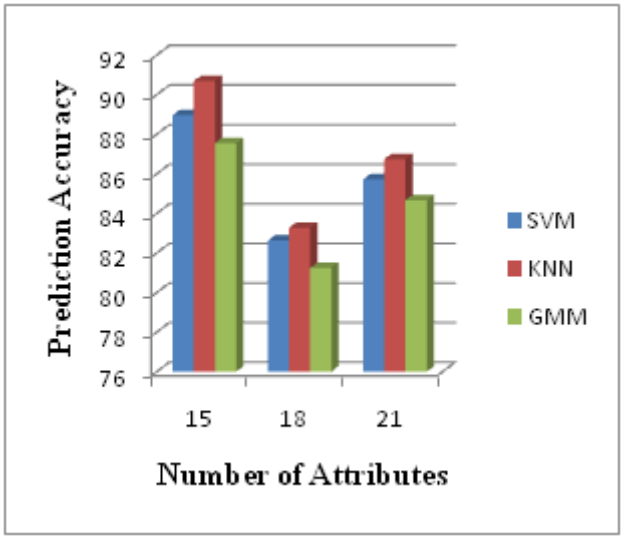

Fig. 3 Comparison of Accuracy for different classifiers

VI. CONCLUSION

The prediction of defects in software is the task of finding defective modules before proceeding with the testing phase. It is hard to eliminate defects completely during the development phase but attempts should be made to keep the defects to a lower count. The prediction of defects will reduce the time and cost required for the development of the software. At the same time it reduces the effort required to rework the module after finding defects during the test phase. Ultimately the defect prediction increases the customer satisfaction and helps developer to build reliable software. Hence, defect prediction become an important practice attain high software quality. This work presents a model based on SVM to overcome this problem. Furthermore, the results of the SVM are compared with the prediction result of the other machine learning techniques. These classifiers can be used to for early prediction of defects in the software components currently uner development. There exist a number of models for

predicting defects in the software but in this work it is concluded that these models depends on the set of software metrics given as input to them.Thus the feature selection plays a major role in increasing the accuracy of the classifiers. The results show that accuracy of the classifiers is increased when selected subset is given as input to them.

REFERENCES

[1] T.J.McCabe, A complexity measure, Software Engineering, IEEE Transactions, vol.4, pp.308-320, 1976.

[2] Halstead, H. Maurice, Elements of Software Science, Elsevier North-Holland, New York,1977.

[3] A. Campan, G. Serban, T. M. Truta, and A. Marcus, An algorithm for the discovery of arbitrary length ordinal association rules, DMIN, vol. 6, pp. 107-113, 2006.

[4] E. E. Mills, Software metrics," 2000.

[5] R. Malhotra, Comparative analysis of statistical and machine learning methods for predicting faulty modules,Applied Soft Computing, vol. 21pp. 286-297 2014.

[6] E. Erturk and E. A. Sezer, A comparison of some soft computing methods for software fault prediction, Expert Systems with Applications, 2014.

[7] M. M. T. Thwin and T.-S. Quah, Application of neural networks for software quality prediction using objectoriented metrics, Journal systems and software, vol. 76, no. 2, pp. 147-156, 2005.

[8] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, Software defect prediction using static code metrics underestimates defect-proneness, in Neural Networks (IJCNN), The 2010 International Joint Conference on, pp. 1-7, IEEE, 2010.

[9] Naidu, M. Surendra, and N. GEETHANJALI. Classification of Defects in Software using Decision Tree Algorithm, International Journal of Engineering Science and Technology (IJEST) 5.06 (2013).

[10] D. Radjenovi_c, M. Heri_cko, R. Torkar, and A. Zivkovi_c, Software fault prediction metrics: A systematic literature review, Information and Software Technology, vol. 55, no. 8, pp. 1397-1418 ,2013.

[11] J. Shirabad, Sayyad, and Tim J. Menzies, The PROMISE repository of software engineering databases, School of Information Technology and Engineering, University of Ottawa, Canada, 2005.

[12] Yang, Jihoon, and Vasant Honavar, Feature subset selection using a geneticalgorithm, Featureextraction, construction and selection, Springer US, pp. 117-136, 1998.