Fault Prediction in Software Modules using Feature Selection and Machine Learning Methods

Varneet Kaur*

Abstract— Software quality assurance is the most important activity during the development of software. Defective software modules may increase costs and decrease customer satisfaction. Hence, effective defect prediction models or techniques are very important in order to deliver efficient software. In this research we are using different machine learning algorithms to predict three main prediction performance measures i.e. precision, recall and f-measure. We are also calculating the accuracy of the software modules. Different classifiers are also used in order to predict the values of these measures by using important attributes only. The results obtained after applying both the techniques i.e. attribute selection and without attribute selection, on all the datasets, are the analysed and best predicted results are chosen in order to predict the correct values of prediction performance measures. The accuracy of some software modules can be improved to 91.16%, recall and precision to 1 after using attribute selection techniques in CM1 dataset. In PC1 dataset the accuracy has been improved to 93.778%.

Index Terms— Defect Prediction Models, Precision, Recall, F-measure, Classifiers

1 INTRODUCTION

Software Quality is the most important aspect during and after the software development. Any defective module in software may lead to increase in its cost and may cause failures and results in customer's dissatisfaction. Delivering a robust, defect free and efficient software is very important; hence there is a huge need of efficient defect prediction models or techniques. Software quality assurance requires both manual inspection and automatic formal methods. There are many modelling techniques that are used in software quality prediction, namely- Discriminant Analysis, Logistic Regression, ANN, Bayes Belief Network, Genetic Algorithms, Classification Trees etc. If a model gives both high defect detection rate and high overall accuracy then it is an efficient and effective defect prediction model.
In this paper we are evaluating 4 NASA datasets [10] namely CM1, JM1, KC1 and PC1. We used many machine learning algorithms available in WEKA, in order to predict modules' precision, recall, f-measure and accuracy. According to Tim Menzies and his colleagues, who worked on JM1, there is a low probability of detecting defective modules [1]
Comparison of many machine learning algorithms on these datasets, performed by Taghi Khoshftaar and Naeem Seliya also predicts low prediction performance [2]. Analysis done by Lan Guo and her colleagues also revealed similar results but they found that Random Forest technique produces better prediction results than other algorithms.[3]
In this paper we introduce a software prediction methodology using different machine learning algorithms. Results obtained by considering all the attributes together are compared with the results obtained by considering attributes after they are ranked by a Ranker algorithm. Many machine learning techniques for attribute selection that have been used in this research are GainRatioAttributeEval, PrincipalComponents, FilteredAttributeEval and ReliefAttributeEval. Different classifiers such as NaiveBayes, BayesNet, SMO, SimpleCart, RandomTree etc. are used in order to predict values of precision, recall, f-measure and accuracy.

1.1 Weka

The Weka workbench is a collection of machine learning algorithms and data preprocessing tools. Weka was developed at the University of Waikato in New Zealand, and the name stands for *Waikato Environment for Knowledge Analysis*. The system is written in Java and distributed under the terms of the GNU General Public License. It runs on almost any platform including Linux, Windows, and Macintosh operating systems. It includes methods for all the standard data mining problems: regression, classification, clustering, association rule mining, and attribute selection. All algorithms take their input in the form of a single relational table in the ARFF format, which can be read from a file or generated by a database query. [7]

2 METHODOLOGY

By using different classifiers and performing cross validation with 10 folds we calculated the commonly used prediction performance measures- Precision, Recall and F-measure. The 10 folds cross validation method partitions the dataset into 10 equal portions, this method uses each portion once as the test set to evaluate the model built using the remaining nine portions. Results obtained by considering all the attributes together are compared with the results obtained by considering attributes after they are ranked by a Ranker algorithm.

Precision: It is the ratio of number of modules correctly predicted as defective to the total number of modules predicted as defective in the set tp+fp.

Here, tp: number of true positives
fp: number of false positives

fn: number of false negatives
tn: number of true negatives

Precision= tp/(tp + fp) **(1)**

Recall: It is the ratio of number of modules predicted correctly as defective to the total number of defective modules in the set tp+fn.
Recall=tp/(tp + fn) **(2)**

F-Measure: It considers precision and recall equally important by taking their harmonic mean. The higher value indicates better prediction performance. [4]
*F-measure = 2 * Recall * Precision / (Recall + Precision)* **(3)**

Fig1: Flowchart of the steps followed (without attribute selection)

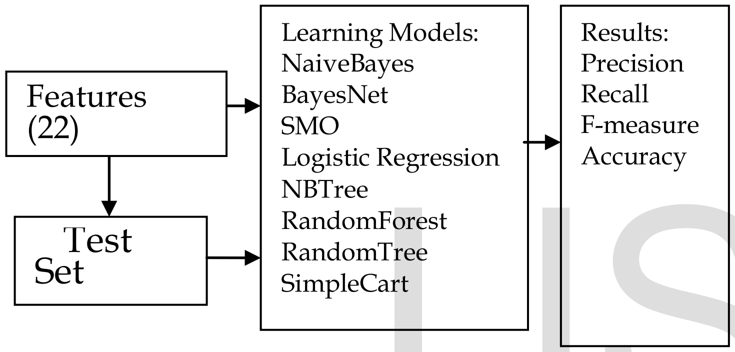
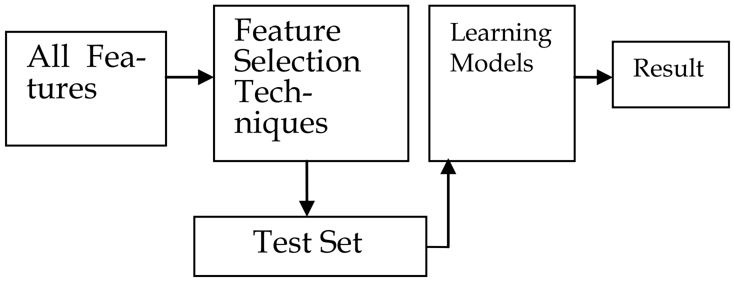


Fig2: Flowchart of the steps followed (with attribute selection)



3 DATASET

In this research we analysed the CM1, JM1, KC1 and PC1 datasets of Promise Repository [5], which belong to 4 software products developed by NASA. In each dataset we analysed the graphs of each attribute. There are 22 attributes, mentioned below:

v(G)-Cyclomatic Complexity, ev(G)- Essential Complexity iv(G)- Design Complexity, LOC- Lines of Code, N- Length V-Volume, L- Level, D- Difficulty, I- Intelligent Count, E- Effort, B- Effort Estimate, T- Programming Time, LOCode- Lines of Code, LOComment- Lines of Comment, LOBlank- Lines of Blank, LOCodeAndComment- Lines of Code and Comment, UniqOp- Unique Operators, UniqOpnd- Unique Operands, TotalOp- Total Operators, TotalOpnd- Total Operands, BranchCount- Total Branch Count

Table 1: Datasets Detail

Project	No. of Modules	Percentage with defects	Language
CM1	496	9.8%	C
JM1	10,885	19.3%	C
KC1	2,109	15.5%	C++
PC1	1,109	6.9%	C

3.1 Classifiers Used

1. **Bayes Net:** It represents the probabilistic dependencies of variables by graph structure.
2. **Naive Bayes:** It is a simple probabilistic classifier based on applying Bayes Theorem with strong independence assumptions. It assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature.
3. **SMO:** SMO chooses to solve the smallest possible optimization problem at every step. It gives highest precision and accuracy table for enemy dataset.[8]
4. **Logistic Regression:** It is useful to predict a dependent variable on the basis of independent variables.[6]
5. **NBTree:** Naive Bayes trees are simply decision trees with Naive bayes classifiers learned from the instances that reach the leaves. All attributes are used in each Naive Bayes model.
6. **Random Trees:** In Random Trees classification works as follows: the random trees classifier takes the input feature vector, classifies it with every tree in the forest, and outputs the class label that received the majority of “votes”. In case of a regression, the classifier response is the average of the responses over all the trees in the forest. All the trees are trained with the same parameters but on different training sets. These sets are generated from the original training set using the bootstrap procedure: for each training set, you randomly select the same number of vectors as in the original set. The vectors are chosen with replacement. That is, some vectors will occur more than once and some will be absent. At each node of each trained tree, not all the variables are used to find the best split, but a random subset of them. With each node a new subset is generated. [11]
7. **Random forests:** Random forests are learning methods for classification that operate by constructing a multitude of decision trees at training time. A random forest is a classifier consisting of a collection of tree-structured classifiers [9].

4. RESULTS AND ANALYSIS

Table 2: Result analysis of CM1 dataset [3]

Method	Precision	Recall	F-measure	Accuracy
BayesNet	0.643	0.947	0.766	64.65%
Naive Bayes	0.910	0.925	0.917	85.34%
Logistic Regression	0.964	0.909	0.936	88.15%
NB Tree	0.979	0.899	0.938	88.35%
Random Tree	0.915	0.911	0.913	84.33%
Random Forest	0.971	0.902	0.935	87.95%

The above table shows the results without any attribute selection. Different classifiers give different results, when attribute selection algorithms are applied to the data set, following results are obtained.

Table 3: Best Results of CM1

Performance Measure	Method	Values	Selection Criteria
Precision	Simple Cart	1	With attribute Selection
Recall	SMO, Simple Cart	1	With Attribute Selection
F-measure	SMO, Simple Cart	0.948	With Attribute Selection
Accuracy	Logistic Regression	89.558%	With Attribute Selection

Table 4: Result Analysis of JM1 [3]

Method	Precision	Recall	F-Measure	Accuracy
BayesNet	1.422	0.876	0.780	68%
Naive Bayes	0.948	0.832	0.886	80.5%
Logistic Regression	0.980	0.822	0.894	81.35%
SMO	1	0.808	0.893	80.72%
Simple Cart	0.977	0.821	0.893	81.11%
NB Tree	0.978	0.822	0.893	81.27%
Random Tree	0.848	0.847	0.848	75.47%
Random Forest	0.948	0.838	0.889	81.05%

The table above gives results without attribute selection. Results vary when some attribute selection techniques are applied.

Table 5: Best Results of JM1

Performance Measure	Method	Values	Selection Criteria
Precision	Bayes Net	1.422	Without attribute Selection
Recall	SMO	1	With Attribute Selection
F-measure	Simple Cart	0.895	With Attribute Selection
Accuracy	Simple Cart, SMO	81.479%	With Attribute Selection

Table 6: Result Analysis of KC1 [3]

Method	Precision	Recall	F-Measure	Accuracy
BayesNet	0.694	0.931	0.796	69.89%
Naive Bayes	0.905	0.888	0.896	82.361%
SMO	0.996	0.849	0.843	84.779%
Simple Cart	0.973	0.863	0.915	84.732%
NB Tree	0.971	0.867	0.916	85.01%
Random Tree	0.913	0.885	0.924	82.693%

The above table shows results without any attribute selection. The following table shows the best results obtained after applying both the method i.e. attribute selection and without attribute selection.

Table 7: Best Results of KC1

Performance Measure	Method	Values	Selection Criteria
Precision	SMO	0.996	Without attribute Selection
Recall	SMO	1	With Attribute Selection
F-measure	Random Tree	0.924	Without Attribute Selection

Accuracy	Random Forest	85.917%	With Attribute Selection

Table 8: Result Analysis PC1 [3]

Method	Precision	Recall	F-Measure	Accuracy
BayesNet	0.759	0.956	0.949	74.391%
Naive Bayes	0.934	0.947	0.940	89.179%
Logistic Regression	0.988	0.934	0.960	92.425%
SMO	0.999	0.930	0.963	92.966%
Simple Cart	0.996	0.935	0.964	93.237%
NB Tree	0.984	0.947	0.965	93.507%
Random Tree	0.950	0.953	0.951	91.073%
Random Forest	0.985	0.946	0.965	93.507%

The above table gives results of without applying any attribute selection algorithms. When attribute selection technique is applied the above results vary. The best results after applying both the techniques are shown below.

Table 9: Best Results of PC1

Performance Measure	Method	Values	Selection Criteria
Precision	SMO	0.999	Without attribute Selection
Recall	SMO	1	With Attribute Selection
F-measure	Random Forest, NBTree	0.967	With Attribute Selection
Accuracy	NBTree	93.778%	With Attribute Selection

5. Conclusion

An important feature of this study is that it has compared results from different machine learning algorithms on several data sets. This paper contributes new results to the framework of software defect prediction. The software prediction performance measures i.e. precision, recall, f-measure and accuracy are calculated on 4 different datasets. These measures have been calculated in two ways firstly by considering all the attributes of the dataset and secondly by attribute selection techniques. The results from these two techniques are compared and the best results are taken into consideration for prediction of faulty modules. The attribute selection techniques in some cases prove to be very efficient and hence improve the prediction performance measures.

In datasets CM1 and KC1 all prediction performance measures values have improved after using attribute selection techniques. Similarly in dataset JM1 recall, f-measure and accuracy has improved when attribute selection is applied. In PC1 dataset the attribute selection techniques give improved values of recall, f-measure and accuracy. Hence, by using attribute selection techniques the accuracy of software modules can be improved to about 93.778%. In this research the attribute selection is done by ranking the attributes using machine learning methods. The higher ranked attributes are selected for calculations. Hence, applying attribute selection techniques provided in Weka tool tends to improve the prediction performance measures and help in effective defect prediction of software modules.

6. References

1. T. Menzies et al., "Mining Repositories to Assist in Project Planning and Resource Allocation," *Proc. 1st Workshop on Mining Software Repositories* (MSR 04), 2004, http://msr.uwaterloo.ca/papers/Menzies.pdf.
2. T.M. Khoshgoftaar and N. Seliya, "The Necessity ofAssuring Quality in Software Measurement Data," *Proc. 10th Int'l Symp. Software Metrics* (METRICS 04),IEEE CS Press, 2004, pp. 119–130.
3. L. Guo et al., "Robust Prediction of Fault Proneness by Random Forests," *Proc. 15th Int'l Symp. Software Reliability Eng.* (ISSRE 04), IEEE CS Press, 2004, pp. 417–428.
4. A.G. Koru and J. Tian, "An Empirical Comparison and Characterization of High Defect and High Complexity Modules," J. Systems and Software, vol.67, no. 3, 2003, pp. 153–163.
5. J.S. Shirabad and T.J. Menzies, "The PROMISE Repository of Software Engineering Databases," School of Information Technology and Engineering, University of Ottawa, Canada, 2005.
6. Lan Guo, Yan Ma, Bojan Cukic, Harshinder Singh, " Robust Prediction of Fault-pronness of Random Forests".
7. Ian H. Witten and Eibe Frank, "Data Mining- Practical Machine learning Tools and Techniques", Second Edition, © 2005 by Elsevier Inc.
8. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, John C. Platt, Microsoft Research jplatt@microsoft.com Technical Report MSR-TR-98-14 April 21, 1998.
9. http://www.stat.berkeley.edu/users/breiman/RandomForests
10. http://promise.site.uottawa.ca/SERepository/datsaets-page.html
11. http://www.stat.berkeley.edu/users/breiman/RandomForests/,http://docs.opencv.org/modules/ml/doc/random_trees.html

***Varneet Kaur**, is a Student, currently pursuing M.E. from the Computer Science Department, Chitkara University, Distt. Solan, Himachal Pradesh, under the guidance of Mr. Amit Arora as guide for the M.E. Thesis.
Email ID: symphony2215@gmail.com