ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 8(35), DOI: 10.17485/ijst/2015/v8i35/73717, December 2015* ISSN (Online) : 0974-5645

An Approach for Software Fault Prediction to Measure the Quality of Different Prediction Methodologies using Software Metrics

R. Sathyaraj[*] and S. Prabu

School of Computing Science and Engineering, VIT University, Vellore - 632014, Tamil Nadu, India; sathyaraj@vit.ac.in, sprabu@vit.ac.in

Abstract

Background: Software fault prediction is an important task to improve the quality of software. It reduces the time and complexity between modules. Future software faults depend on previous faulty data. **Statistical Analysis:** The prediction models are created using method level metrics available from NASA datasets of structure oriented CM1 and PC1 datasets and object oriented KC1 and KC2 datasets. These metrics were applied in different classifier like Naive Bayes, J48, K-Star and Random forest to identify the best classifier for small dataset and large dataset based on both structure and object oriented method level metrics. **Findings:** This paper gives the study and analysis of various methodologies used for predicting faults in both structure and object oriented software. Based on the study, Naïve Bayes is utmost suitable for small datasets and random forest is suitable for large datasets based on the evaluation done by us using various methodologies driven by WEKA tool while equating precision, recall and accuracy. **Application:** This process of software fault prediction can be applied to E-Commerce applications, where accuracy requires much importance.

Keywords: Naive Bayes, Random Forest, Software Metrics, Software Quality, Software Fault Prediction

1. Introduction

Prediction is an important process in software development, which avoids any distraction in software process and improves the quality, also reduce time complexity. Most of the fault prediction models used previous datasets to predict fault. These prediction models benefits to improve the design approach by classifying alternative approach for fault models and also improving quality[1]. In software development, we can identify whether the software development process is in right path or not by measuring the changes happening in the development process. To take concrete decision in software development quality these changes facilitated mostly[2]. Software success depends upon the quality, grounded by exact working of the product for which it was made[3].

In this paper four classifiers (Random forest, J48, Lazy K-Star and Naive Bayes) had taken and compared to explicit uniqueness and finest for prediction using NASA open dataset. The NASA structure oriented CM1 and PC1 and object oriented KC1 and KC2 were used as datasets and the classifiers Random forest, J48, Lazy K-Star and Naive Bayes were classified using WEKA tool to find out the accuracy in prediction. Based on the results acquired, Random forest was suitable for large dataset and Naïve Bayes was suitable for small dataset.

2. Related Works

Cagatal Catal and Banu Diri (2009) identified high performance fault prediction algorithm based on machine learning. This study used NASA data and made predictive models with re-usability. Study acknowledged seven test groups and nine algorithms were executed in

** Author for correspondence*

each dataset. This study investigates the LOC; metrics set and used feature selection techniques. Out of five dataset, only one performed class level metrics, so three groups were worked on this class metrics. Experiment results also shown that valid hypotheses were HP1, HP2, HP3, HP4, and HP7 plus invalid were HP5 and HP6. They concluded Random forest is best for large dataset and Naive Bayes for small dataset. In these study AIRS2 parallel algorithm, a new computational paradigm AIS based prediction was best for method level metrics and Immunos2 algorithm was best for class level metrics[4].

Jiang et al. discussed strength and weakness of performance evaluation techniques, also noticed cost characteristics of project has to consider for selecting best model. They used NASA MDP thirteen datasets and six classification algorithms to demonstrate the best algorithm with cost effect. The classifiers were Random forest, Naive Bayes, Logistic regression, Bagging, J48 and IBK. This study strongly mentions the use of cost curve provided standard for performance evaluation of software quality model and Random forest performance was best[5].

Supreet Kaur and Dinesh Kumar expresses DBSCAN (Density Based Spatial Clustering of Application with Noise) evaluated java based object oriented systems with accurately. In this study, they exercised metric based approach for prediction and also they weighed the performance of fault proneness of classes using NASA metrics data program data repository for C++ language based components. A result shown reduced set of attributes exactness of prediction was increased from 85.3712% to 90.6114%, so metrics based approach was suitable in the reduced set of attributes. DBSCAN result showed the precision of prediction of 58.63% for the reduced set attributes. So study shows DBSCAN was suitable for prediction based on fault proneness[6].

Julie Moeyersoms et al. study concentrated software project effort needed to complete and fault prediction tried to identify faulty modules using various data mining methodologies. In their study they used Random forest and SVM (Support Vector Machine) for regression making use of rule extraction algorithm ALPA. Based on Android repository, public dataset admired with new datasets applied to proposed methodology. Results shown, that the tree extracted from black box and how the prediction created from black box models[7].

Boetticher examined the effects of datasets on software engineering and analysed NASA public datasets using J48 and Naive Bayes techniques. PD, PF and accuracy were the performance evaluation metrics. In this paper, the datasets were divided into three parts: training set, nice neighbours test set, and nasty neighbours test set. This study showed that for nice neighbours test set accuracy was 94% and for nasty neighbours test set accuracy was 20%. Also stated, for validation ten-fold cross-validation was not sufficient and if the evaluation was critical when the datasets were difficult[8].

3. Methodology

The following footsteps label the functionalities in this paper,

- Dataset chosen based on the size from NASA MDP Promise repository.
- Popular four classifiers have chosen. They were Random forest, J48, K-Star and Naive Bayes.
- Using WEKA tool these four classifiers analyze the dataset and find correctly and incorrectly classified instances in each dataset.
- Find the precision, recall and accuracy of each classifier.
- Finally analyze the result and prompt which algorithm is best for what type of dataset.

3.1 Dataset Used

Software fault prediction is important feature in improving quality of software development. For prediction different metrics were used in various research works. Dataset may be public or private; here public dataset can be used by everyone and used for many applications. But private dataset cannot be accessed and used by all and it has some issues and procedures to use those datasets.

Promise software engineering repository datasets:

For these studies the data collected from NASA Metrics Data program datasets CM1, PC1, KC1 and KC2[9].

3.1.1 CM1 Dataset

CM1/Software defect prediction creator was NASA metrics data program. It was a NASA spacecraft instrument, written in C language. Features of McCabe and Halstead metrics extracted from source code. These metrics were segment based or it may call as function or method. CM1 has 498 numbers of instances.

3.1.2 KC1 Dataset

KC1/Software defect prediction creator was NASA metrics data program. It was storage management for receiving and processing ground data project, written in C++ language. Features of McCabe and Halstead metrics extracted from source code. KC1 has 2109 numbers of instances.

3.1.3 KC2 Dataset

KC2/Software defect prediction creator was NASA metrics data program. It was a science data processing; it was another portion of KC1 project, written in C++ language. It added third party software libraries with KC1. Data originates from McCabe and Halstead metrics extracted from source code. KC2 has 522 numbers of instances.

3.1.4 PC1 Dataset

PC1/Software defect prediction creator was NASA metrics data program. It was flight software designed for earth orbiting satellite, written in C language. Data originates from McCabe and Halstead metrics extracted from source code. PC1 has 1109 numbers of instances

3.2 Classifiers

3.2.1 Naive Bayes

Since 1950, Naive Bayes has been considered for prediction and it is most important classifier in prediction. Naive Bayes classifiers are used in learning problems because it is highly accessible. Naïve Bayes classifier is working based on Bayes theorem. It built independent probability model. It is working in many real world problems[10]. It needs small dataset quantity to classify the problem.

3.2.2 J48

J48 is in WEKA data mining tool, implementation of C4.5 algorithm. It is a Java open source implementation for decision making. It generates tree based data output for set of input values. Ross Quinlan was developed this algorithm. It handles continuous and discrete attributes. It is suitable for fault prediction in any size of datasets.

3.2.3 Random Forest

Random forest is a collection of decision trees, presented independently with certain controlled modification. It includes many trees and the result based on majority of accurate output chosen the class. Random forest is the best classifier for large datasets.

3.2.4 K-Star

K star algorithm is an instance based classifier and uses entropy based distance functionality. Although it is based on instance classifier but it is differentiated by using entropic measure. It works based on similar methods will have related classification. This methodology little bit slower to evaluate but good for prediction.

3.3 Factors Considered for Calculating Performance of Classifiers

Classifiers accuracy acquired by, true positive rate, false positive rate, precision, recall and F-measures using WEKA tool. WEKA tool is a powerful collection of machine learning algorithms for classification, clustering, regression, and for all data mining tasks.

3.3.1 True Positive (TP)

True positive, proportion categorized as class x / Actual total in class x. True positive projected by the modules that are predicted positively as the results specified at the end. For example, if the person identified as healthy, then the results shown as it is, then it is a true positive.

True Positive rate = True Positive / (True Positive + False Negative)

3.3.2 False Positive (FP)

False positive, proportion incorrectly categorized as class x / Actual total of all classes, except x. It is incorrectly predicted compared to original results. For example, if the person identified as unhealthy, then the results shown as healthy, then it is a false positive.

False Positive rate = False Positive / (False Positive + True Negative)

3.3.3 Precision

Precision proportion of the examples which truly have class x / Total classified as class x. Precision gives positive predictive values and it process values or product quality or exactness. So basically high precision stated the accurate results and it takes all relevant data but returns only topmost results. In short, "just see how many chosen items were related".

Precision = True Positive / (True Positive + False Positive)

3.3.4 Recall

Recall gives sensitivity of problem and it process values or product quantity or completeness. It returned most relevant and part of the documents that are relevant as result from the query. In other words, modules that are really recognize as difficult to maintain from the total number of modules. In short, "just see how many related objects were chosen".

Recall = True Positive / (True Positive + False Negative)

3.3.5 F-Measure

F-Measure categorized as 2*Precision*Recall / (Precision + Recall). It is a combined measure for precision and recall.

3.3.6 Accuracy

Accuracy calculated as number of instances predicted positively divided by Total number of instances. In the experiment the values of the accuracy posted into table in the basis of 0 to 1, not from 0 to 100.

Accuracy = (True Positive + True Negative) / (P + N)

4. Experiments and Results

4.1 Experiment

In this study, 21 metrics form Mccabe's and Halstead metrics and one goal metric were taken to measure. Using Weka tool, CM1, KC2 and KC1, PC1 datasets were applied to classifiers Naive Bayes, J48, K-Star and Random forest algorithms. These dataset has taken combination in basis of structure and object oriented. CM1 and PC1 were written in C and KC1 and KC2 were written in C++ language.

Study had compared average of accuracy (the values taken in to table as from 0 to 1), true positive rate, false positive rate, precision, recall and F-measures. For better performance, we can also go for area of ROC curve values. Accuracy calculated based on number of instances classified correctly. Based on these analysis results, Naive Bayes methodology was suitable for small dataset and Random Forest classifier was suitable for large dataset.

4.2 Results

Tables 1, 3, 5 and 7 have given the instances correctly classified and in-accurately classified with total number instances in dataset using different classifiers. Tables 2, 4, 6 and 8 listed the true positive rate, false positive rate, precision, recall and F-measures to analyse the classifiers. Also it provides best classifier by highlighted based on precision value.

4.2.1 Structure Oriented Small Dataset - CM1

Table 1. Classified instances of CM1 dataset

Method	Appropriately Classified Instances	Inaccurately Classified Instances	Total Instances
Naive Bayes	425	73	498
J48	438	60	498
Random forest	442	56	498
K-Star	434	64	498

Table 2. Accuracy analysis on CM1 dataset

	CM1					
	TP rate	FP rate	Preci sion	Recall	F-Measure	Accuracy
NB	0.853	0.616	0.862	0.853	0.858	0.83
J48	0.88	0.849	0.833	0.88	0.852	0.88
RF	0.88	0.867	0.832	0.88	0.854	0.89
K-Star	0.87	0.705	0.857	0.871	0.863	0.87

Figure 1 describe the ratio of each classifier for each dataset based on Table 2.

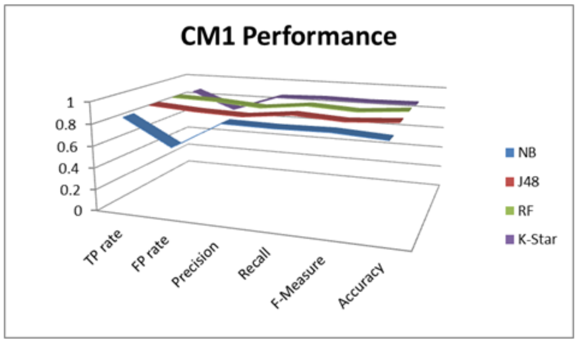

Figure 1. Performance analysis on CM1 dataset.

4.2.2 Object Oriented Small Dataset – KC2

Table 3. Classified Instances of KC2 dataset

Method	Appropriately Classified Instances	Inaccurately Classified Instances	Total Instances
Naive Bayes	436	86	522
J48	425	97	522
Random forest	432	90	522
K-Star	413	109	522

Table 4. Accuracy analysis on KC2 dataset

	KC2					
	TP rate	FP rate	Preci sion	Recall	F-Measure	Accuracy
NB	0.835	0.473	**0.820**	0.835	0.821	0.84
J48	0.814	0.422	0.807	0.814	0.810	0.81
RF	0.828	0.440	0.815	0.828	0.819	0.83
K-Star	0.791	0.498	0.778	0.791	0.783	0.79

Figure 2 describe the ratio of each classifier for each dataset based on Table 4.

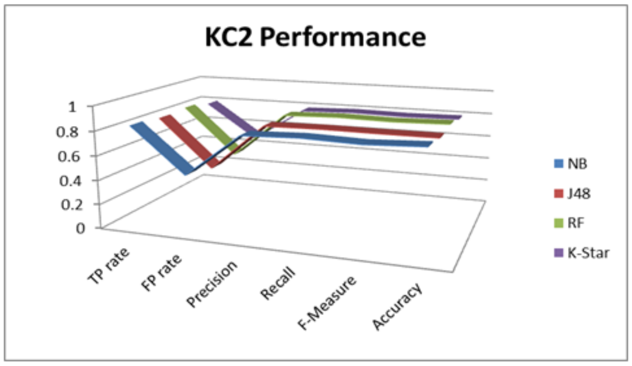

Figure 2. Performance analysis on KC2 dataset.

4.2.3 Structure Oriented Large Dataset - PC1

Table 5. Classified instances of PC1 dataset

Method	Appropriately Classified Instances	Inaccurately Classified Instances	Total Instances
Naive Bayes	989	120	1109
J48	1035	74	1109
Random forest	1402	67	1109
K-Star	1018	91	1109

Table 6. Accuracy analysis on PC1 dataset

	PC1					
	TP rate	FP rate	Preci sion	Recall	F-Measure	Accuracy
NB	0.982	0.657	0.899	0.892	0.895	0.89
J48	0.933	0.714	0.917	0.933	0.921	0.93
RF	0.94	0.653	**0.928**	0.94	0.929	0.94
K-Star	0.918	0.691	0.906	0.918	0.911	0.92

Figure 3 describe the ratio of each classifier for each dataset based on Table 6.

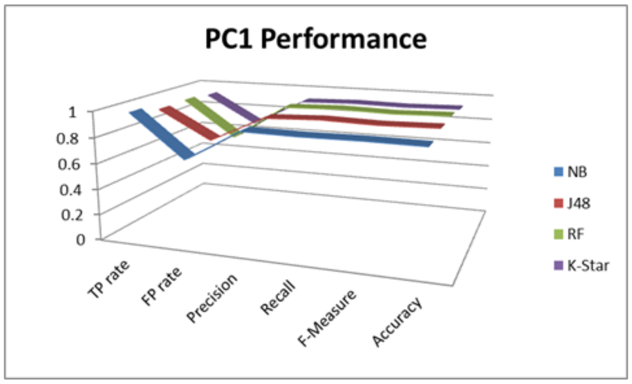

Figure 3. Performance analysis on PC1 dataset.

4.2.4 Object Oriented Large Dataset – KC1

Table 7. Classified Instances of KC1 dataset

Method	Appropriately Classified Instances	Inaccurately Classified Instances	Total Instances
Naive Bayes	1737	372	2109
J48	1783	326	2109
Random forest	1821	288	2109
K-Star	1771	338	2109

Table 8. Accuracy analysis on KC1 dataset

	KC1					
	TP rate	FP rate	Preci sion	Recall	F-Measure	Accuracy
NB	0.824	0.541	0.816	0.824	00.82	0.82
J48	0.845	0.575	0.825	0.845	0.832	0.85
RF	0.863	0.576	**0.843**	0.863	0.845	0.86
K-Star	0.840	0.538	0.826	0.84	0.832	0.84

Figure 4 describe the ratio of each classifier for each dataset based on Table 8.

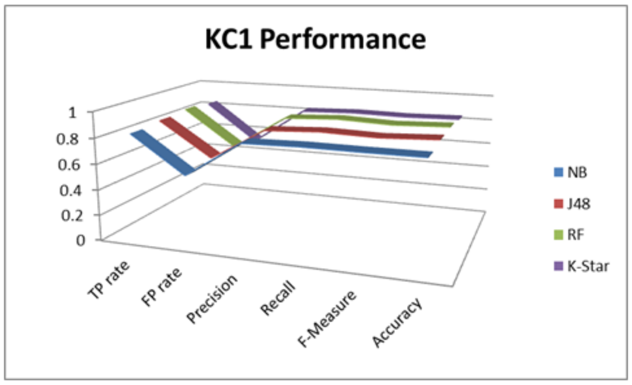

Figure 4. Performance analysis on KC1 dataset.

5. Conclusion

The main objective of this work is to analyse the performance of different classifiers using different metrics of NASA datasets. This study uses the measures like true positive rate, false positive rate, precision, recall and F-measures to analyse the classifiers. Based on the study and analysis of various methodologies it has been proved that Naive Bayes is suitable for small datasets and random forest is suitable for large datasets. The future enhancement of this proposed work has been planned to do the same process for Java based open source projects for same metrics and planned to extend for E-Commerce networking projects.

6. References

1. Catal C. Software fault prediction: A literature review and current trends. Experts Systems with Applications. 2011; 38(4):4626-36.
2. Rashid, Ekbal, Patnayak S, Bhattacherjee V. Estimation and evaluation of change in software quality at a particular stage of software development. Indian Journal of Science and Technology. 2013; 6(10):5370-9.
3. Sathyaraj R, Prabu S. A survey-quality based object oriented software fault prediction. International Journal of Engineering and Technology. 2013 Jun–Jul; 5(3): 2349-51.
4. Catal C, Diri B. Investigating the effect of dataset size, metrics sets and feature selection techniques on software fault prediction problem. Information Sciences. 2009; 170(8):1040-58.
5. Jiang Y, Cukic B, Ma Y. Techniques for evaluating fault prediction models. Empirical Software Eng. 2008; 13(5):561–95.
6. Kaur S, Kumar D. Software fault prediction in object oriented software systems using density based clustering approach. International Journal of Research in Engineering and Technology (IJRET). 2012 Mar; 1(2):111-7.
7. Moeyersoms J, Fortuny EJ, Dejaeger K, Baesens B. Comprehensible software fault and effort prediction: A data mining approach. The Journal of Systems and Software. 2015 Feb; 100:80-90.
8. Boetticher G. Improving credibility of machine learner models in software engineering. Advanced Machine Learner Applications in Software Engineering. Hershey, PA, USA: Idea Group Publishing; 2006.
9. NASA Metrics Data Program. 2015 Apr 15. Available from: http://promise.site.uottawa.ca/SERepository/datasets-page.html
10. Catal C, Sevim U, Diri D. Practical development of an Eclipse-based software fault prediction tool using Naive Bayes algorithm. Expert Systems with Applications. 2011; 38(3):2347-53.