Bagged SVM Classifier for Software Fault Prediction

Shanthini. A
Research scholar,
Department of Computer Science and Engineering,
Annamalai University,
Annamalai nagar,
Tamil Nadu, India

Vinodhini.G
Assistant professor,
Department of Computer Science and Engineering,
Annamalai University,
Annamalai nagar ,
Tamil Nadu, India

Chandrasekaran.RM
Professor,
Department of Computer Science and Engineering,
Annamalai University,
Annamalai nagar,
Tamil Nadu, India

ABSTRACT

Defective modules in the software pose considerable risk by decreasing customer satisfaction and by increasing the development and maintenance costs. Therefore, in software development life cycle, it is essential to predict defective modules in the early stage so as to improve software developers' ability to identify the defect-prone modules and focus quality assurance activities. Many researchers focused on classification algorithm for predicting the software defect. On the other hand, classifiers ensemble can effectively improve classification performance when compared with a single classifier. This paper mainly addresses using ensemble approach of Support Vector Machine (SVM) for fault prediction. Ensemble classifier was examined for Eclipse Package level dataset and NASA KC1 dataset. From the research, it is clear that proposed ensemble of Support Vector Machine is superior to individual approach for software fault prediction in terms of classification rate through Root Mean Square Error Rate (RMSE), Area Under ROC Curve (AUC-ROC) and Area Under Precision and Recall curve (AUC-PR).

Keywords

Defect prediction, Software metrics, Machine learning, Class level metrics, Method level metrics.

1. INTRODUCTION

Faults and failures in software are costly factors. They account for a significant amount of any project budget. This cost cannot be removed completely as methods are needed to ensure the quality of the software. The costs for fault handling should be possible to decrease considerably by introducing and improving methods for early fault identification. Thus, fault prediction can be used for process improvement and hence cost reduction. Based on the above reasoning, it is clear that methods are needed to predict, control and improve fault handling in general [6]. Thus, methods for identification of fault-and failure prone modules and models for fault prediction are a potential way to improve software quality and to reduce cost. Effective fault prediction methods on defect prone modules can help developers to focus quality on assurance activities and thus improve software quality by using resources more efficiently. The fault prediction methods often use metrics obtained from source code. The metrics includes size, coupling, cohesion, inheritance, and complexity metrics which have been associated with risk factors, such as defects and changes [7][10].

Data Mining has become a very powerful technique to reduce information overload and improve decision making by extracting and refining useful knowledge from extensive dataset through a process of searching for relationships and patterns. In recent years data mining techniques have been successfully used in software fault prediction. There are many data mining methods used to predict the faults in the modules [4] [8] [9] [6]. The primary objective of this paper is to show that ensemble of Support Vector Machine is superior to individual machine learning approach. The rest of this paper is organized as follows. Subsequent sections describe related work. The 3rd section is for the data source. The 4th section present details about metrics and machine learning approaches used. Performance evaluation is discussed in the 5th section. Conclusions are given in the 6th section.

2. REVIEW OF RELATED LITERATURE

Considerable research has been performed on software metrics and defect prediction models. Catal et al. [3] examined Chidamber-Kemerer metrics suite and some method-level metrics (the McCabe's and Halstead's ones) for a defect model which is based on Artificial Immune Recognition System (AIRS) algorithm. The authors investigated together 84 metrics from the method-level metrics transformation and 10 metrics from the class-level metrics. According to obtained results the authors concluded that the best fault prediction is achieved when CK metrics are used with the lines of code (LOC) metric.

In [2], the author has used various machine learning techniques for an intelligent system for the software maintenance prediction and proposed the logistic model Trees (LMT) and Complimentary Naïve Bayes (CNB) algorithms on the basis of Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Accuracy percentage.

Menzies et al. [4] showed that Naive Bayes with logNums filter provides the best performance on NASA datasets for software fault prediction problem.

Olague et al. [5] found that the complexity metrics have a good performance in distinguishing between fault-prone and not fault-prone classes. In addition, they also found that lesser known metrics such as SDMC and AMC were better predictors than the commonly used metrics LOC and WMC.

Elish et al. [8] stated that the performance of Support Vector Machines (SVM) is generally better than, or at least is competitive against the other statistical and machine learning models in the context of four NASA datasets.

The use of Machine Learning for the purpose of predicting or estimating software module's fault-proneness is proposed by [9], which views fault-proneness as both a continuous measure and a binary classification task. Using a NASA public dataset, a NN is used to predict the continuous measure while a SVM is used for the classification task. Gondra's [9]

experimental results showed that Support Vector Machines provided higher performance than the Artificial Neural Networks for software fault prediction.

Kanmani et al. [6] validated Probabilistic Neural Network (PNN) and Back propagation Neural Network (BPN) using a dataset collected from projects of graduate students, in order to compare their results with results of statistical techniques. According to Kanmani et al.'s [6] study, PNN provided better performance.

3. FAULT PREDICTION DATASET

The data set used in this research is obtained from the NASA IV & V Facility Metrics Data Program (MDP) data repository. The primary objective of the MDP is to collect, validate, organize, store and deliver software metrics data. The repository contains software metrics and associated error data for several projects. The data is made available to general users and has been sanitized by officials representing the projects from which the data originated. For each project in the database, unique numeric identifiers are used to describe product entries. A product refers to anything with which defect data and metrics can be associated. In most cases, it refers to code-related project modules such as functions. For each module, metric values were extracted and mapped to a defect log. Because the recorded metric values for a module correspond to those obtained before eliminating faults in the module, there is no risk of the metric values changing as a result of structural changes in the module that may occur during fault elimination. In this research, the data associated with the KC1 project is considered for evaluation. This is a real-time project written in C++ consisting of approximately 315,000 LOC. There are 10,878 modules and 145 classes.

Another data set used in this research is obtained from the bug database of Eclipse. The data set lists the number of pre- and post-release defects for every package in the Eclipse 3.0. All data is publicly available and used for defect prediction models. Dataset consists of following attributes. Each case contains the following information:

n**ame:** The name of the file or package, respectively ,to which this case corresponds. It can be used to identify the source code in the release and may be needed for additional data collection.

p**re-release defects:** The number of non-trivial defects that were reported in the last six months before release.

p**ost-release defects:** The number of non-trivial defects that were reported in the first six months after release.

c**omplexity metrics:** Metrics that are computed for classes or methods are aggregate by using average (avg), maximum (max), and accumulation (sum) to package level.

s**tructure of abstract syntax tree(s):** For each case, we list the size (=number of nodes) of the abstract syntax tree(s) of the package, respectively.

3. CLASSIFICATION METHODS

3.1 SVM

Support vector machine (SVM) is also well known tool for performing data classification, and have been successfully used in many applications. SVM constructs an N-dimensional hyper plane that optimally separates the data set into two categories. The purpose of SVM modeling is to find the optimal hyper plane that separates clusters of vector in such a way that cases with one category of the dependent variable on one side of the plane and the cases with the other category on the other side of the plane. The support vectors are the vectors near the hyper plane. The SVM modeling finds the hyper plane that is oriented so that the margin between the support vectors is maximized. When the points are separated by a nonlinear region, SVM handles this by using a kernel function in order to map the data into a different space when a hyper plane can be used to do the separation[8],[9],[6].

4.2 Ensemble SVM

Ensemble learning techniques have been shown to increase machine learning accuracy by combining arrays of specialized learners. Bagging and boosting [7] are examples of ensemble methods. Bagging is a "bootstrap" ensemble method that creates individuals for its ensemble by training each classifier on a random redistribution of the training set.

Algorithm: Bagging.

The bagging algorithm creates an ensemble of models (classifiers or predictors) for a learning scheme where each model gives an equally weighted prediction.

Input:

D, a set of d training tuples;

K, the number of models in the ensemble;

A learning scheme (e.g., decision tree algorithm, back propagation, etc.)

Output: A composite model, M.*

Method:

(1) for i = 1 to k do // create k models:

(2) create bootstrap sample, Di, by sampling D with replacement;

(3) use Di to derive a model, Mi;

(4) end for

To use the composite model on a tuple, X:

(1) if classification then

(2) let each of the k models classify X and return the majority vote;

(3) if prediction then

(4) let each of the k models predict a value for X and return the average predicted value

Several researchers have investigated the combination of different classifiers to from an ensemble classifier. An important advantage for combining redundant and complementary classifiers is to increase robustness, accuracy and better overall generalization. In this approach, the Support Vector Machine is constructed and 10-fold cross validation technique is applied and evaluated error rate from the mean square error. Secondly, bagging is performed with Support Vector Machine to obtain a very good generalization performance.

5. EXPERIMENTS

In this research paper, the WEKA machine learning library is used as the source of algorithms for experimentation. The bagging and SVM classification algorithms are implemented in WEKA with default parameters.

5.1 Root Mean Square Error Rate

RMSE is frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modelled or estimated. It is just the square root of the mean square error as shown in equation given below.

Assuming that the actual output is a, expected output is c.

$$\sqrt{\frac{(a_1 - c_1)^2 + (a_2 - c_2)^2 + \ldots + (a_n - c_n)^2}{n}}$$

The data set described in section 2 is being used to test the performance of Ensemble of Support Vector Machine. Root Mean square error (RMSE) was evaluated using 10-fold cross validation as cross validation is the best technique to get a reliable error estimate. Root Mean square error was evaluated using ensemble Method. The Root Mean square error in the figure 1 reflects the best performance of bagging with Support Vector Machine in terms classification rate. The error can be reduced to zero as the number of classifiers combined to infinity.

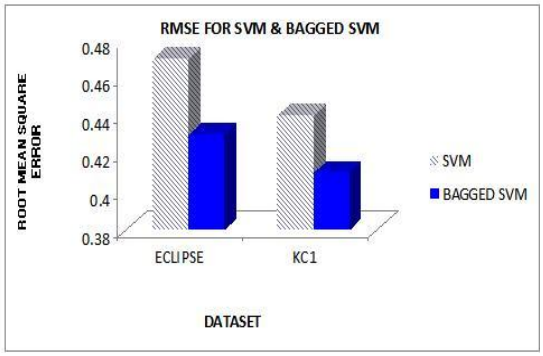

Figure. 1. RMSE for SVM & Bagged SVM

5.2 Area Under Curve (AUC)

The data set described in section 2 is being used to test the performance of bagging with Support Vector Machine. The ROC curve in this experiment is used to evaluate the performance of ensemble algorithm. AUC-ROC is used as a performance metrics (area under ROC curve), an integral of ROC curve with false positive rate as x axis and true positive rate as y axis. If ROC curve is more close to top-left of coordinate, the corresponding classifier must have better generalization ability so that the corresponding AUC will be lager. Therefore, AUC can quantitatively indicate the generalization ability of corresponding classifier.

Figure 2 and 3 shows the ROC curves for class 0 and class 1, evaluating the performance curve of SVM and Bagged SVM classifier on the eclipse data set (JAVA). Figure 4 and 5 shows the Area under ROC curves evaluating the performance curve of SVM and bagged SVM classifiers on the KC1 data set (C++).

Receiver Operator Characteristic curves are commonly used to present results for binary decision problems in machine learning. However, when dealing with highly skewed datasets, Precision-Recall (PR) curves give a more informative picture of an algorithm's performance. We show that a deep connection exists between ROC space and PR space, such that a curve dominates in ROC space if and only if it dominates in PR space. A ROC and PR curves are typically generated to evaluate the performance of a machine learning algorithm on a given dataset. Figure 6 and Figure 7 shows the precision and recall curve for Eclipse and KC1 data set for class 0 and class 1. Area under the ROC curves and Precision Recall curve (AUC-ROC and AUC-PR) are calculated and the results were shown in the table 1. From the AUC values (AUC-ROC and AUC-PR) it is evident that, for both the data set (Eclipse data set and KC1 data set) Bagged SVM classifier gives better performance.

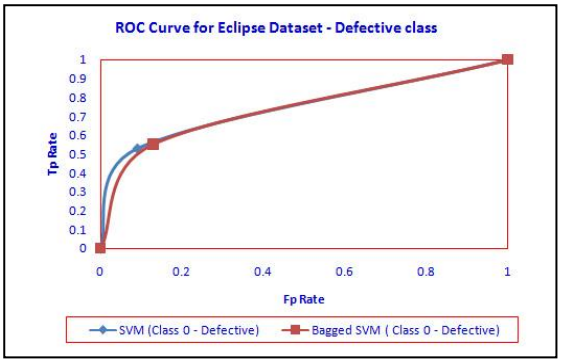

Figure. 2. ROC for Eclipse dataset (class 0)

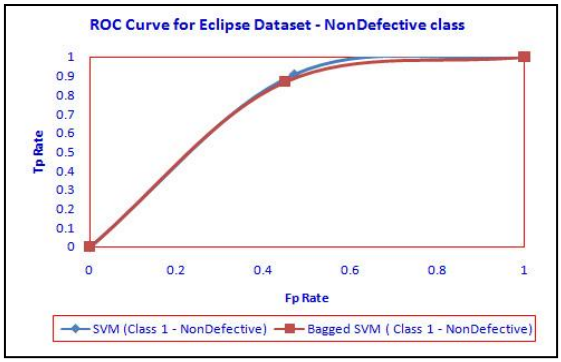

Figure. 3. ROC for Eclipse dataset (class 1)

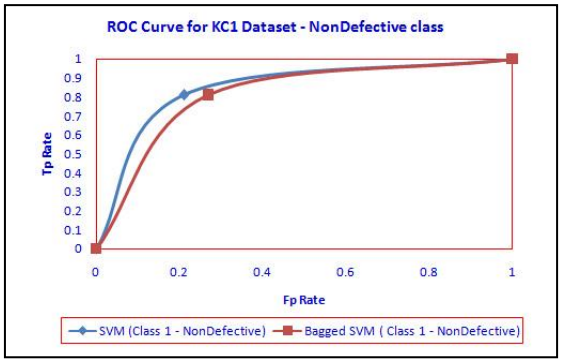

Figure. 4. ROC for KC1 dataset (class 0)

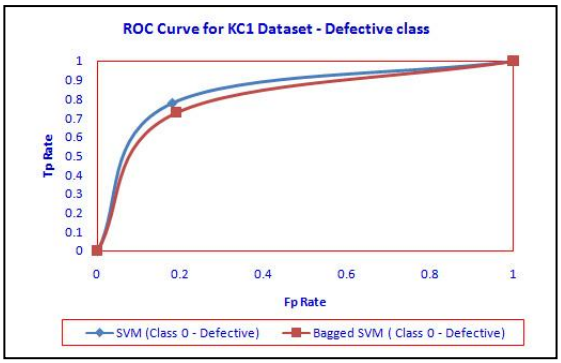

Figure. 5. ROC for KC1 dataset (class 1)

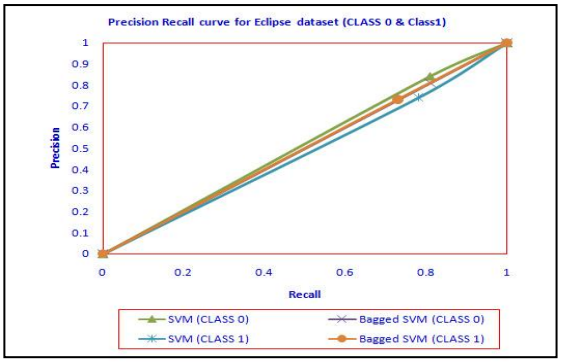

Figure. 6. PR curve for Eclipse dataset (class 0& class1)

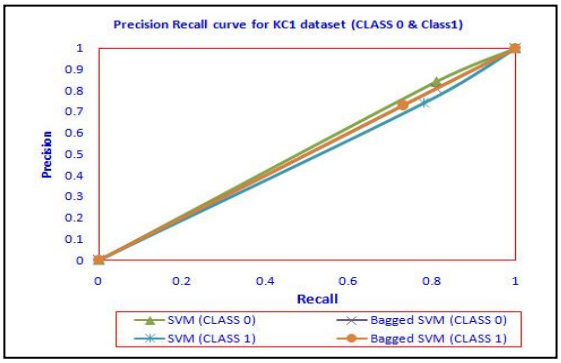

Figure. 7. PR curve for KC1 dataset (class 0& class1)

Table 1. Area under the curve (AUC- ROC and AUC-PR)

CLASSIFIER	AUC-ROC (Eclipse)	AUC-PR (Eclipse)	AUC-ROC (KC1)	AUC-PR (KC1)
SVM (CLASS 0)	0.721	0.58	0.798	.67
Bagged SVM (CLASS 0)	0.78	0.70	0.832	0.84
SVM (CLASS 1)	0.721	0.761	0.798	0.79
Bagged SVM (CLASS 1)	0.78	0.81	0.832	0.73

6. CONCLUSIONS

The goal of this research is to analyze the performance of various classifiers for various metrics level data set on defect prediction. The performance of the classifiers using Root Mean Square Error is analyzed. Roc is also used as an alternative metric. A ROC and PR curves are typically generated to evaluate the performance of a machine learning algorithm on a given dataset. The area under the ROC curves (AUC-ROC) and (AUC–PR) are calculated by using the trapezoidal method. From the ROC curves (AUC-ROC and AUC-PR) it is evident that, for both the data set (Eclipse data set and KC1 data set) Bagged SVM classifier gives better performance. Many researchers apply machine learning methods for constructing the model to predict faulty classes. The extension of this research is to predict the models based on other machine learning algorithms such as ensemble using neural networks and genetic algorithms.

7. REFERENCES

[1] Knab, P., Pinzger, M., and Bernstein, A., 2006. "Predicting defect densities in source code files with decision tree learners," in the 2006 International Workshop on Mining Software Repositories.

[2] Sandhu, Parvinder Singh, Sunil Kumar and Hardeep Singh, 2007 "Intelligence System for Software Maintenance Severity Prediction",Journal of Computer Science, Vol. 3 (5), pp. 281-288.

[3] Catal, C., Diri, B., and Ozumut, B., 2007. "An Artificial Immune System Approach for Fault Prediction in Object-Oriented Software," in 2nd International Conference on Dependability of Computer Systems DepCoS-RELCOMEX.

[4] Menzies, T., Greenwald, J., & Frank, A. (2007). Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering, 33(1),

[5] Olague, H.M., Etzkorn, L.H., Gholston, S., Quattlebaum, S., 2007. Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. IEEE Transactions on Software Engineering 33 (6), 402– 419.

[6] S.Kanmani, V.R. Uthariaraj, V.Sankaranarayanan, P.Thambidurai, Objected-oriented software fault prediction using neural networks, Information and software Technology 49 (5 (2007)) 483 – 492.

[7] Dr Kadhim M. Breesam, "Metrics for Object Oriented design focusing on class Inheritance metrics", 2nd International conference on dependability of computer system IEEE, 2007.

[8] K.O. Elish, M.O. Elish, Predicting defect-prone software modules using support vector machines, Journal of Systems and Software 81 (5) (2008) 649– 660.

[9] I.Gondra, Applying machine learning to software fault-proneness prediction, Journal of System and Software 81(2) (2008) 186-195.

[10] Amjan Shaik, Dr C.R.K. Reddy, Dr A Damodaran, "Statistical Analysis for Object Oriented Design Software security metrics", International journal of engineering and technology, vol. 2, pg 1136-1142,2010

[11] Cagatay Catal, "Software fault prediction: A literature review and current trends", Expert Systems with Applications 38 (2011) 4626 – 4636.