Determination of Fault Proneness of Modules in Open Source Software Systems using SVM Clustering Approach

Ritika* , Bhupinder Singh2 and Satinder Pal Ahuja3

Abstract—There are available metrics for predicting fault prone classes, which may help software organizations for planning and performing testing activities. Fault-proneness of a software module is the probability that the module contains faults. This may be possible due to proper allocation of resources on fault prone parts of the design and code of the software. Hence, importance and usefulness of such metrics is understandable, but empirical validation of these metrics is always a great challenge. SVM algorithm has been successfully applied for solving regression and classification problems in many applications. This paper evaluates the capability of SVM algorithm in predicting fault prone software classes using open source software. The results indicate that the Support vector machine method to predict the faulty classes with better accuracy.

Keywords— Software metrics, fault prediction, machine learning, Support vector machine Approach, software quality.

I. INTRODUCTION

Prediction of fault-prone modules provides one way to support software quality engineering through improved scheduling and project control. Quality of software is increasingly important and testing related issues are becoming crucial for software. Although there is diversity in the definition of software quality, it is widely accepted that a project with many defects lacks quality. Methodologies and techniques for predicting the testing effort, monitoring process costs, and measuring results can help in increasing efficiency of software testing. Being able to measure the fault-proneness of software can be a key step towards steering the software testing and improving the effectiveness of the whole process. Predictive modeling is the process by which a model is created or chosen to try to best predict the probability of an outcome. The objective of a fault-proneness model is to identify faulty classes and focus testing effort on them.

II. SOFTWARE METRICS

Metrics are designed to provide information about the software quality and allow easier detection of possible error or bad design.

Metrics used in the classification process are:
- Weighted methods per class (WMC)
- Depth of inheritance tree (DIT)
- Number of Children (NOC)
- Coupling Between Object Classes (CBO)
- Lack of Cohesion in Methods (LCOM)

III. INTRODUCTION TO MACHINE LEARNING TECHNIQUES

The major focus of machine learning research is to extract information from data automatically, by computational and statistical methods. Hence, machine learning is closely related to data mining and statistics. Machine Learning is part of Machine Intelligence but addresses a more specialized purpose and scope. Machine learning algorithms and applications adapt themselves to the behavior of a system usually through the discovery of time-varying patterns in the data.

IV. PROBLEM FORMULATION

There are many metrics and technique available for investigate the accuracy of fault prone classes which may help software organizations for planning and performing testing activities. As the complexity and the constraints under which the software is developed are increasing, it is difficult to produce software without faults. Such faulty software classes may increase development & maintenance cost, due to software failures and decrease customer’s satisfaction. A variety of software fault predictions techniques have been proposed, but none has proven to be consistently accurate. These techniques include statistical method, machine learning methods, parametric methods and mixed algorithms. Therefore, there is a need to find the best prediction techniques for a given prediction problem.

V. ALGORITHM USED

SVM Clustering based prediction system. SVM is a useful technique for data classification. Even though it’s considered that Neural Networks are easier to use than this, however, sometimes unsatisfactory results are obtained. A classification task usually involves with training and testing data which
consist of some data instances. Each instance in the training set contains one target value and several attributes. The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes.

Support Vector Machines (SVM) Introductory Overview

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. A schematic example is shown in the illustration below. In this example, the objects belong either to class GREEN or RED. The separating line defines a boundary on the right side of which all objects are GREEN and to the left of which all objects are RED. Any new object (white circle) falling to the right is labeled, i.e., classified, as GREEN (or classified as RED should it fall to the left of the separating line).

The above is a classic example of a linear classifier, i.e., a classifier that separates a set of objects into their respective groups (GREEN and RED in this case) with a line. Most classification tasks, however, are not that simple, and often more complex structures are needed in order to make an optimal separation, i.e., correctly classify new objects (test cases) on the basis of the examples that are available (train cases). This situation is depicted in the illustration below.

Compared to the previous schematic, it is clear that a full separation of the GREEN and RED objects would require a curve (which is more complex than a line). Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.

The illustration below shows the basic idea behind Support Vector Machines. Here we see the original objects (left side of the schematic) mapped, i.e., rearranged, using a set of mathematical functions, known as kernels. The process of rearranging the objects is known as mapping (transformation). Note that in this new setting, the mapped objects (right side of the schematic) is linearly separable and, thus, instead of constructing the complex curve (left schematic), all we have to do is to find an optimal line that can separate the GREEN and the RED objects.

Support Vector Machine (SVM) is primarily a classier method that performs classification tasks by constructing hyper planes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables. For categorical variables a dummy variable is created with case values as either 0 or 1. Thus, a categorical dependent variable consisting of three levels, say (A, B, C), is represented by a set of three dummy variables:

A: \{1 0 0\}, B: \{0 1 0\}, C: \{0 0 1\}

To construct an optimal hyper plane, SVM employees an iterative training algorithm, this is used to minimize an error function. According to the form of the error function, SVM models can be classified into four distinct groups:

Classification SVM Type: For this type of SVM, training involves the minimization of the error function:

\[
\frac{1}{2}w^Tw + C\sum_{i=1}^{N}\xi_i
\]

subject to the constraints:

\[
y_i\left(w^T\phi(x_i) + b\right) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0, \quad i = 1, ..., N
\]

where C is the capacity constant, w is the vector of coefficients, b a constant and \(\xi_i\) are parameters for handling nonseparable data (inputs). The index i labels the N training cases. Note that \(y_i \in \{\pm 1\}\) is the class labels and xi is the independent variables. The kernel \(\phi\) is used to transform data from the input (independent) to the feature space. It should be noted that the larger the C, the more the error is penalized. Thus, C should be chosen with care to avoid over fitting.

The performance criterion taken is the classification Accuracy\%. It is the percentage of the predicted values that match with the expected values of the reusability for the given data. The best system is that having the high Accuracy value.

VI. RESULTS AND DISCUSSION

The data is collected from the statistics of the metric data of the WMC, DIT, NOC, CBO, RFC, LCOM, NPM, LOC metrics is tabulated in Table metrics respectively. The detail of the number of Faulty and Non-Faulty Modules present in the dataset is shown in Table 1.
### TABLE I
**STATICS OF THE WMC METRIC VALUES IN JEDIT DATA**

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td>Maximum</td>
<td>407</td>
</tr>
<tr>
<td>Mean</td>
<td>11.725</td>
</tr>
<tr>
<td>StdDev</td>
<td>91.202</td>
</tr>
</tbody>
</table>

### TABLE II
**STATICS OF THE DIT METRIC VALUES IN JEDIT DATA**

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td>Maximum</td>
<td>7</td>
</tr>
<tr>
<td>Mean</td>
<td>2.496</td>
</tr>
<tr>
<td>StdDev</td>
<td>1.977</td>
</tr>
</tbody>
</table>

### TABLE III
**STATICS OF THE NOC METRIC VALUES IN JEDIT DATA**

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td>Maximum</td>
<td>35</td>
</tr>
<tr>
<td>Mean</td>
<td>0.715</td>
</tr>
<tr>
<td>StdDev</td>
<td>3.1</td>
</tr>
</tbody>
</table>

### TABLE IV
**STATICS OF THE CBO METRIC VALUES IN JEDIT DATA**

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td>Maximum</td>
<td>105</td>
</tr>
<tr>
<td>Mean</td>
<td>12.642</td>
</tr>
<tr>
<td>StdDev</td>
<td>14.131</td>
</tr>
</tbody>
</table>

### TABLE V
**STATICS OF THE RFC METRIC VALUES IN JEDIT DATA**

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td>Maximum</td>
<td>843</td>
</tr>
<tr>
<td>Mean</td>
<td>174.978</td>
</tr>
<tr>
<td>StdDev</td>
<td>269.591</td>
</tr>
</tbody>
</table>

### TABLE VI
**STATICS OF THE LCOM METRIC VALUES IN JEDIT DATA**

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td>Maximum</td>
<td>100</td>
</tr>
<tr>
<td>Mean</td>
<td>46.237</td>
</tr>
<tr>
<td>StdDev</td>
<td>33.516</td>
</tr>
</tbody>
</table>

### TABLE VII
**STATICS OF THE NPM METRIC VALUES IN JEDIT DATA**

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>0</td>
</tr>
<tr>
<td>Maximum</td>
<td>193</td>
</tr>
<tr>
<td>Mean</td>
<td>7.785</td>
</tr>
<tr>
<td>StdDev</td>
<td>17.121</td>
</tr>
</tbody>
</table>
First the training and testing dataset is generated from the JEdit dataset using “Generate cross-validation indices” utility. Thereafter the training of is performed using SVM algorithm. After the training, the training structure is created. Thereafter, the testing is performed on the basis of testing dataset already created. When SVM in the MATLAB environment is executed on the dataset then the following results are obtained:

```
Label: ' '
Description: ''
ClassLabels: [2x1 double]
GroundTruth: [274x1 double]
NumberOfObservations: 274
ControlClasses: 2
TargetClasses: 1
ValidationCounter: 1
SampleDistribution: [274x1 double]
ErrorDistribution: [274x1 double]
SampleDistributionByClass: [2x1 double]
ErrorDistributionByClass: [2x1 double]
CountingMatrix: [3x2 double]
CorrectRate: 0.7810
ErrorRate: 0.2190
LastCorrectRate: 0.7810
LastErrorRate: 0.2190
InconclusiveRate: 0
ClassifiedRate: 1
Sensitivity: 0.7857
Specificity: 0.7761
PositivePredictiveValue: 0.7857
NegativePredictiveValue: 0.7761
PositiveLikelihood: 3.5095
NegativeLikelihood: 0.2761
Prevalence: 0.5109
DiagnosticTable: [2x2 double]
```
The Accuracy of best classification is calculated as 78.1022%, mean the error rate is 0.2190 as shown in the figure above.

This paper empirically evaluates performance of Support vector machine (SVM) technique in predicting fault-prone classes using open source software. Here, SVM structure generated from fault data in MATLAB 7.4 environment is evaluated for the JEdit dataset. The proposed SVM based prediction technique shows 78.1022 percent Accuracy as best case means it has outperformed the Random forest based prediction system which has reported 76.2774 accuracy in literature.

Hence, this study confirms that construction of SVM based model is feasible and useful in predicting faulty prone classes. It is therefore concluded that, in case of open source software, model is implemented using SVM based technique for classification of the software components into faulty/fault-free systems is found satisfactory. The contributions of the study can be summarized as follows: First open source software systems analyzed. These systems are developed with different development methods than proprietary software. In previous studies mostly proprietary software were analyzed. Second, we examine Support vector machine method to predict the faulty classes with better accuracy.

The future work can be extended in following directions:

- Most important attribute can be found for fault prediction

and this work can be extended to further programming languages.
- More algorithms can be evaluated and then we can find the best algorithm. We plan to replicate our study to predict model based on hybrid genetic algorithms or soft computing techniques.

REFERENCES


