A *k*-means Based Approach for Prediction of Level of Severity of Faults in Software System

Jaspreet Kaur, Parvinder S. Sandhu

Abstract—Fault-proneness of a software module is the probability that the module contains faults. A correlation exists between the fault-proneness of the software and the measurable attributes of the code (i.e. the static metrics) and of the testing (i.e. the dynamic metrics). Early detection of fault-prone software components enables verification experts to concentrate their time and resources on the problem areas of the software system under development. Among various clustering techniques available in literature K-means clustering approach is most widely being used. This paper introduces K-means based Clustering approach for software finding the fault proneness of the Object-Oriented systems. The contribution of this paper is that it has used Metric values of JEdit open source software for generation of the rules for the categorization of software modules in the categories of Faulty and non faulty modules and thereafter empirically validation is performed. The results are measured in terms of accuracy of prediction, probability of Detection and Probability of False Alarms.

Keywords— *K*-means, Software Fault, Classification, Object Oriented Metrics.

I. INTRODUCTION

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. Prediction of fault-prone modules provides one way to support software quality engineering through improved scheduling and project control. 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. Faults in software systems continue to be a major problem. Many systems are delivered to users with excessive faults. This is despite a huge amount of development effort going into fault reduction in terms of quality control and testing. It has long been recognized that seeking out fault-prone parts of the system and targeting those parts for increased quality control and testing is an effective approach to fault reduction. Fault-proneness of a software module is the probability that the module contains faults. A correlation exists between the fault-proneness of the software and the measurable attributes of the code (i.e. the static metrics) and of the testing (i.e. the dynamic metrics). 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. Methodologies and techniques for predicting the testing effort, monitoring process costs, and measuring results can help in increasing efficiency of software. The following are the metrics used in the classification process:

- Coupling between Objects (CBO)
- Lack of Cohesion (LCOM)
- Number of Children (NOC)
- Depth of inheritance (DOI)
- Weighted Methods per Class (WMC)
- Response for a class (RFC)
- Number of Public Methods (NPM)
- Lines of Code (LOC)

Thereafter, the reduced number of attributes are given as input to the *k*-means clustering algorithm. As Clustering is technique that divides data in to two or more clusters depending upon some criteria. As, in this study data is divided in to two clusters depending upon that whether they are fault free or fault prone. In the *k*-means technique Euclidian distance as well as Manhattan distance measures are experimented. If the Manhattan distance is used, then centroids are computed as the component-wise median rather than mean.

To predict the results, we have used confusion matrix. The confusion matrix has four categories: True positives (TP) are the modules correctly classified as faulty modules. False positives (FP) refer to fault-free modules incorrectly labeled as faulty. True negatives (TN) are the fault-free modules correctly labeled as such. False negatives (FN) refer to faulty modules incorrectly classified as fault-free modules.

The following set of evaluation measures are being used to find the results:

- Probability of Detection (PD), also called recall or specificity, is defined as the probability of correct classification of a module that contains a fault.

\[ PD = \frac{TP}{(TP + FN)} \]  

- Probability of False Alarms (PF) is defined as the ratio of false positives to all non defect modules.
\[ PF = \frac{FP}{FP + TN} \]  

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. In the past, several metrics for measuring software complexity and testing thoroughness have been proposed. Static metrics, e.g., the McCabe's cyclomatic number or the Halstead's Software Science, statically computed on the source code and tried to quantify software complexity. Despite this it is difficult to identify a reliable approach to identifying fault-prone software components.

Clustering is used to determine the intrinsic grouping in a set of unlabeled data. It is the process of organizing objects into groups whose members are similar in some way. Among various clustering techniques available in literature \textit{k-means} clustering approach is most widely being used. \textit{k-means} is an unsupervised clustering technique used to classify data into \textit{k} clusters. It is partitional clustering approach, each cluster is associated with a centroid (center point), each point is assigned to the cluster with the closest centroid. Number of clusters, \textit{K}, must be specified. Hence, in this study, a \textit{k-means} approach is used for detecting faulty Modules in Open Source Software Systems. In order to perform the analysis we validate the performance of the \textit{k-means} based clustering method for dataset derived from open source software JEdit [1]. We investigate the accuracy of the fault proneness predictions using object oriented design using metrics suite given by Chidamber and Kemerer [2] and used in [3] for fault prediction. In the literature [3]-[17] various types of Fault-Proneness Estimation Models are discussed.

II. METHODOLOGY USED

In course of the research work, the following steps will be required:

- Study of the metrics needed for maintenance.
- Collect the sampled relevant metric data.
- Analyze and refine metrics data.
- Use the following algorithm of \textit{k-means} Clustering based prediction system:

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{flowchart.png}
\caption{Flowchart of \textit{k-means} algorithm}
\end{figure}

Step 1. Begin with a decision on the value of \textit{k} = number of clusters.

Step 2. Put any initial partition that classifies the data into \textit{k} clusters. You may assign the training samples randomly, or systematically as the following:

- Take the first \textit{k} training sample as single-element clusters.
- Assign each of the remaining (\textit{N}-\textit{k}) training sample to the cluster with the nearest centroid. After each assignment, recomputed the centroid of the gaining cluster.

\textit{Step 3.} Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.

\textit{Step 4.} Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

If the number of data is less than the number of cluster then we assign each data as the centroid of the cluster. Each centroid will have a cluster number. If the number of data is bigger than the number of cluster, for each data, we calculate the distance to all centroid and get the minimum distance. This data is said belong to the cluster that has minimum distance from this data.

Deduce the results on basis of accuracy, precision and recall values. In case of the two-cluster based problem, the confusion matrix has four categories: True positives (TP) are modules correctly classified as faulty modules. False positives (FP) refer to fault-free modules incorrectly labeled as faulty modules. True negatives (TN) correspond to fault-free modules correctly classified as such. Finally, false negatives (FN) refer to faulty modules incorrectly classified as fault-free modules as shown in table 3.1.

<table>
<thead>
<tr>
<th>Predicted Value of Level of severity of faults</th>
<th>Real Data Value of Level of severity of faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TP</td>
</tr>
<tr>
<td>0</td>
<td>FN</td>
</tr>
</tbody>
</table>

With help of the confusion matrix values the precision and recall values are calculated described below:

A. Precision

The Precision is the proportion of the examples which truly have class \textit{x} among all those which were classified as class \textit{x}. The technique having maximum value of probability of...
detection and lower value of probability of false alarms is chosen as the best fault prediction technique.

Precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). The equation is:

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

(3)

B. Recall

Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been) [8]. The recall can be calculated as follows:

\[
\text{Recall} = \frac{TP}{TP + FN}
\]

(4)

C. Accuracy

The percentage of the predicted values that match with the expected values for the given data is called Accuracy%.

The best system is that having the high Accuracy, High Precision and High Recall value.

III. Math

The data is collected from [1] and the statistics of the metric data of the WMC, DIT, NOC, CBO, RFC, LCOM, NPM, LOC metrics is tabulated in Table II, III, IV, V, VI and IX metrics respectively. The details of the number of Faulty and Non-Faulty Modules present in the dataset.

<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>487</td>
</tr>
<tr>
<td>Mean</td>
<td>11.728</td>
</tr>
<tr>
<td>StDev</td>
<td>32.082</td>
</tr>
</tbody>
</table>

TABLE II
STATISTICS 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>1135</td>
</tr>
<tr>
<td>Mean</td>
<td>12.762</td>
</tr>
<tr>
<td>StDev</td>
<td>14.131</td>
</tr>
</tbody>
</table>

TABLE III
STATISTICS 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>5</td>
</tr>
<tr>
<td>Mean</td>
<td>2.4586</td>
</tr>
<tr>
<td>StDev</td>
<td>1.3777</td>
</tr>
</tbody>
</table>

TABLE IV
STATISTICS 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>5</td>
</tr>
<tr>
<td>Mean</td>
<td>6.05</td>
</tr>
<tr>
<td>StDev</td>
<td>4.43</td>
</tr>
</tbody>
</table>

TABLE V
STATISTICS 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>640</td>
</tr>
<tr>
<td>Mean</td>
<td>174.578</td>
</tr>
<tr>
<td>StDev</td>
<td>268.551</td>
</tr>
</tbody>
</table>

TABLE VI
STATISTICS OF THE RFC METRIC VALUES IN JEDIT DATA

<table>
<thead>
<tr>
<th>Search Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best first:</td>
</tr>
<tr>
<td>Start set no attributes</td>
</tr>
<tr>
<td>Search direction: forward</td>
</tr>
<tr>
<td>Start search after 5 node expansions</td>
</tr>
<tr>
<td>Total number of subsets evaluated: 43</td>
</tr>
<tr>
<td>Worst of best subset found: 0.229</td>
</tr>
</tbody>
</table>

Attribute Subset Evaluator (supervised, Class (nominal), 9 Bug-count)

Selected attributes: 2, 4, 5, 7, 8, 9

DIT

CBO

RFC

NPM

LOC

Fig. 2 Snapshot of the Output of Correlation-based Feature Subset Selection using BestFirst Search

The parameters used are:

- direction -- Set the direction of the search. The default value ‘forward’ is used.
- lookupCacheSize -- Set the maximum size of the lookup cache of evaluated subsets. This is expressed as a multiplier of the number of attributes in the data set. It is set to 1.
- searchTermination -- Set the amount of backtracking. It is set to 5.
The figure 2 shows the results after applying Correlation based Feature Subset Selection using BestFirst Search. It has proposed the use of DIT, CBO, RFC, NPM and LOC metrics significant metrics for the prediction. In case of Chi-squared Ranking Filter selection ranks attributes by their individual evaluations. Use in conjunction with attribute evaluators (ReliefF, GainRatio, Entropy etc). The following parameters are used:

- distanceFunction -- The distance function to use for instances comparison. First set to Euclidean Distance and thereafter set to Manhattan Distance.
- dontReplaceMissingValues -- Replace missing values globally with mean/mode. Default value False is used.
- maxIterations -- set maximum number of iterations. It is set to 500.
- numClusters -- set number of clusters. It is set to 2.
- preserveInstancesOrder -- Preserve order of instances. It is set to False.
- seed -- The random number seed to be used. It is set to 10.

The proposed $K$-means based classification technique shows 62.4 percent accuracy. It also shows high value of Probability of detection (PD) i.e. 0.754 and low value of Probability of False Alarms (PF) i.e. 0.413.

This study confirms that construction of $K$-means based model is feasible, adaptable to Object Oriented systems and useful in predicting faulty prone classes. It is therefore concluded that model is implemented using $K$-means 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 $K$-means clustering 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


