Predicting the Impact of the Defect on the Overall Environment in Function Based Systems

Parvinder S. Sandhu, Urvashi Malhotra, and Ebru Artil

Abstract—There is lot of work done in prediction of the fault proneness of the software systems. But, it is the severity of the faults that is more important than number of faults existing in the developed system as the major faults matters most for a developer and those major faults needs immediate attention. In this paper, we tried to predict the level of impact of the existing faults in software systems. Neuro-Fuzzy based predictor models is applied NASA’s public domain defect dataset coded in C programming language. As Correlation-based Feature Selection (CFS) evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. So, CFS is used for the selecting the best metrics that have highly correlated with level of severity of faults. The results are compared with the prediction results of Logistic Models (LMT) that was earlier quoted as the best technique in [17]. The results are recorded in terms of Accuracy, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). The results show that Neuro-fuzzy based model provide a relatively better prediction accuracy as compared to other models and hence, can be used for the modeling of the level of impact of faults in function based systems.

Keywords—Software Metrics, Fuzzy, Neuro-Fuzzy, Software Faults, Accuracy, MAE, RMSE.

I. INTRODUCTION

In the literature [1] [2], [3], [4], [5], [6] made prediction of fault prone modules in software development process and mostly used the metric based approach with machine learning techniques to model the fault prediction in the software modules. Khoshgoftaar [7] used zero-inflated Poisson regression to predict the fault-proneness of software systems with a large number of zero response variables. Munson and Khoshgoftaar [8, 9] also investigated the application of multivariate analysis to regression and showed that reducing the number of “independent” factors (attribute set) does not significantly affect the Accuracy of software quality prediction. Menzies, Ammar, Nikora, and Stefano [10] compared decision trees, naïve Bayes, and 1-rule classifier on the NASA software defect data. Emam, Benlarbi, Goel, and Rai [11] compared different case-based reasoning classifiers and concluded that there is no added advantage in varying the combination of parameters (including varying nearest neighbor and using different weight functions) of the classifier to make the prediction Accuracy better. Many modeling techniques have been developed and applied for software quality prediction [12], [13], [14], [15]. The software quality may be analyzed with limited fault proneness data [16].

In [17], 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.

Neuro-Fuzzy systems have proven to be of great practical value in a variety of application domains especially in case of non linear modeling of systems. In this paper, we tried to predict the level of impact of the existing faults in software systems. Neuro-Fuzzy based predictor models are applied NASA’s public domain defect dataset coded in C programming language. The results are compared with the prediction results of Logistic Models (LMT) that was earlier quoted as the best technique in [17].

This paper is organized as: Section two describes the Methodology part of work done, which shows the steps used in order to reach the objectives and carry out the results. In the section three, results of the implementation are discussed. In the last section, on the basis of the discussion various Conclusions are drawn and the future scope for the present work is discussed.

II. PROPOSED METHODOLOGY

A. Find the Structural Code and Design Attributes

The first step is to find the structural code and design attributes of software systems i.e. software metrics that are relevant in modeling the impact of severity of faults in function oriented systems. The real-time defect data sets are taken from the NASA’s MDP (Metric Data Program) data repository. The dataset is related to the safety critical software systems being developed by NASA.
B. Collection of Metric Values

The suitable metrics values of different product modules is collected and observed severity of faults in those modules is noted.

C. Refinement of Metrics using CFS

A central problem in machine learning is identifying a representative set of features from which to construct a classification model for a particular task. In the step the metrics are refined using Correlation based Feature Selection (CFS) technique. This CFS addresses the problem of feature selection for machine learning through a correlation based approach. The central hypothesis is that good feature sets contain features that are highly correlated with the class. A feature evaluation formula, based on ideas from test theory, provides an operational definition of this hypothesis. In other words, Correlation-based Feature Selection evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them.

D. Experimentation with Neuro-Fuzzy based System

The Neuro-Fuzzy based system is used in prediction of level of impact of faults. Neural Network is used to train the Sugeno based Fuzzy Inference System. The results are calculated in terms of the different criteria explained in the next point.

E. Comparison Criteria

The comparisons of machine learning algorithms are made on the basis of following criteria:

- **Accuracy**
  The percentage of the predicted impact values that match with the expected values of the level impact of faults in the modules.

- **Mean Absolute Error**
  Mean absolute error, $MAE$ is the average of the difference between predicted and actual value in all test cases; it is the average prediction error [21]. The formula for calculating $MAE$ is given in equation shown below:

$$
MAE = \frac{1}{n} \sum_{i=1}^{n} |a_i - c_i| 
$$

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

- **Root Mean-Squared Error**
  $RMSE$ is frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated [21]. It is just the square root of the mean square error as shown in equation given below:

$$
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (a_i - c_i)^2} 
$$

The best system is that having the highest **Accuracy** and least values of **MAE** and **RMSE**.

F. Conclusions Drawn

The conclusions are made on the basis of the comparison made in the previous section.

III. RESULTS & DISCUSSION

The real-time defect data set used is taken from the NASA’s MDP (Metric Data Program) data repository, the details of that dataset contains 178 modules of C Programming language with different values of software fault severity labeled as 1, 2 and 3. The level 1 represents the highest severity, level 2 represents the medium and level 3 represents the minor fault that can be overlooked to save time. Details of the type of faults existing in different number of modules of the Dataset are shown in bar chart of Fig. 1.

There are total 21 metrics in the dataset of NASA. The detail of the metrics is shown in the following table:

<table>
<thead>
<tr>
<th>Metric Type</th>
<th>Metric</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>v(G)</td>
<td>Cyclomatic Complexity</td>
<td>(CC)</td>
</tr>
<tr>
<td>McCabe</td>
<td>Essential Complexity(EC)</td>
<td>Design Complexity(DC)</td>
</tr>
<tr>
<td>n(G)</td>
<td>Lines of Code Executable</td>
<td></td>
</tr>
<tr>
<td>ELOC</td>
<td>Length</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>Volume</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>Level</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Difficulty</td>
<td></td>
</tr>
<tr>
<td>Derived</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Halstead</td>
<td>Intelligent Count</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>Effort</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Effort Estimate</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Programming Time</td>
<td>Lines of Code</td>
</tr>
<tr>
<td>T</td>
<td></td>
<td>Lines of Comment</td>
</tr>
<tr>
<td>Line</td>
<td></td>
<td>Lines of Blank</td>
</tr>
<tr>
<td>LOCcode</td>
<td></td>
<td>Lines of Code and Comment</td>
</tr>
<tr>
<td>Count</td>
<td></td>
<td>UniqOp</td>
</tr>
<tr>
<td>LOCAndComment</td>
<td></td>
<td>Unique Operators</td>
</tr>
<tr>
<td>Basic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UniqOpnd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Halstead</td>
<td>TotalOp</td>
<td>Total Operators</td>
</tr>
<tr>
<td>TotalOpnd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Branch</td>
<td>TotalBranchCount</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1 Graphical Representation of Details of the Type of Modules in the Dataset

TABLE I

Details of the Metric Group of NASA Data [17]
In order to incorporate the correlation of independent variables, a correlation based feature selection technique (CFS) is applied to select the best predictors out of independent variables in the datasets [19]. The best combinations of independent variable are searched through all possible combinations of variables. CFS evaluates the best of a subset of variables by considering the individual predictive ability of each feature along with the degree of redundancy between them [20]. The following are the five metrics that are further used for the NF based modeling:

- **LOC_CODE_AND_COMMENT**: The number of lines which contain both code and comment in a module.
- **HALSTEAD_LENGTH**: The Halstead length metric of a module.
- **NUM_OPERANDS**: The number of operands contained in a module.
- **NUM_OPERATORS**: The number of operators contained in a module.
- **LOC_TOTAL**: The total number of lines for a given module.

Given separate sets of input and output data, subtractive clustering algorithm is used to generate a Fuzzy Inference system (FIS). The inputs and output of the FIS are shown in Fig. 2. It is accomplished by extracting a set of rules that models the data behavior. The rule extraction method first determines the number of rules and antecedent membership functions and then uses linear least squares estimation to determine each rule's consequent equations. This function returns an FIS structure that contains a set of fuzzy rules to cover the feature space.

![Fig. 2 Fuzzy Inference System Generated using Subtractive Clustering](image)

<table>
<thead>
<tr>
<th>Radii Value</th>
<th>Performance Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td>0.15</td>
<td>71.91</td>
</tr>
<tr>
<td>0.17</td>
<td>72.47</td>
</tr>
<tr>
<td>0.2</td>
<td>67.41</td>
</tr>
<tr>
<td>0.25</td>
<td>66.85</td>
</tr>
</tbody>
</table>

During the testing phase the performance of the NF in predicting the Impact of Faults at different Radii Values is noted down as shown in the Table I. The proposed system shows optimal performance at radii 0.17 with 66.85, 0.454 and 0.605 as Accuracy (%), MAE and RMSE values.

In literature [17] has mentioned that Logistic Model Trees (LMT) is best among other machine learning algorithms used in that study for the prediction of maintenance severity. So, Logistic Model Trees (LMT) is applied on the polished dataset it has shown Accuracy, MAE and RMSE values as 56.74, 0.2269, and 0.334 respectively.

IV. CONCLUSION

According to [17], on comparing all the classes of WEKA’s machine learning algorithms, it is observed that Logistic Model Trees algorithm is best prediction techniques as compared with other classes of machine learning algorithms in prediction of severity of faults in software systems. But, the proposed Adaptive Neuro-fuzzy based prediction technique has outperformed Logistic Model Trees technique on the basis of the testing data with 66.85, 0.454 and 0.605 as Accuracy, Mean Absolute Error and Root Mean Square Error values.

It is therefore, concluded the model is implemented and the
best algorithm for classification of the software components into different level of severity of impact of the fault is found to be Neuro-Fuzzy-based technique. The algorithm can be used to develop models that can be used for identifying modules that are heavily affected by the faults and those can be debugged at appropriate time.

REFERENCES


