CHAPTER 4

INVESTIGATION OF SOFTWARE DEFECT PREDICTION

4.1 Business Intelligence Model Using Data Mining Techniques for Code Optimization in Legacy Systems.

Legacy systems are older software systems that are still functional, running on outdated hardware and typically its original designers and implementers are no longer available to perform the system's maintenance [6]. The legacy systems which are using obsolete c are still used, even though new technologies which are more efficient are available, as its data or application programs cannot be upgraded. Often documentation is not available or lacks clarity, so the only definitive source of information about the system is the code itself.

4.1.1 Introduction

Reasons for maintaining a legacy system:

- The system is still functional and is able to meet the user’s need. The system cannot be taken out of service, because a new system costs higher for a similar feature level. The systems handling customers' accounts in banks, computer reservation, air traffic control, energy distribution (power grids), nuclear
power plants and military defense installations are some examples of the legacy systems.

- The working of the system is not clear. This kind of situation occurs when the designers of the system have left the organization, and the system documentation has been lost [13].

- Incorporating newer systems is difficult because new software is not compatible as it may use different technologies. The clash of software and hardware, when incorporating newer software will necessitate a different hardware. Thus the cost of upgrading becomes prohibitive.

“An approach to cope with the legacy systems problem is reengineering which addresses the problem in two parts. The first part, called reverse engineering, focuses on understanding the legacy system and the second part, called forward engineering, uses the information produced in the reverse engineering stage and adds new specifications to rebuild the legacy system using modern technologies” [47]. A relevant sub area of reverse engineering is identification of modules requiring reengineering, which focuses on predicting faulty modules. To this end, predicting faulty modules using any existing source of information like documentation and source code. Similarly, the identified generalizations may take different forms such as module breakdown, structure-charts, entity-relationship diagrams, and software metric specifications. Predicting defective module plays an
essential role in maintenance and reuse; by simplifying the workings of the system with information and the localization of reusable parts. As suggested in the previous chapter, there are many approaches to the module defect prediction. It differs in many ways such as the extraction procedures and tools employed, the sources of information used, and the outcome produced. In this chapter, the classification accuracy of Random tree, Bayesian Logistic Regression and CART based on the KC1 dataset is investigated. Data mining is the process of extracting patterns or models from data by applying specific algorithms [42].

4.1.2 Receiver Operating Characteristics (ROC)

Receiver Operating Characteristics (ROC) graphs are used for evaluating and comparing algorithms. It is a useful performance graphing technique which depicts the tradeoff between true positive rates and false alarm rates of the classifier [59, 60]. ROC is widely used in machine learning, as it is useful for domains with skewed class distribution and unequal classification error costs. ROC curves also quantify the overall effectiveness of the various algorithms used in a particular study.

In a classification problem, for a classifier and an instance, there are four possible outcomes. The possible outcomes are:
Instance is positive and classified as positive – True Positive (TP)

Instance is positive and classified as negative – False Negative (FN)

Instance is negative and classified as negative – True Negative (TN)

Instance is negative and classified as positive – False Positive (FP)

For a set of instances, confusion matrix is constructed representing the dispositions of the instances. This matrix forms the basis for many common matrix. The true positive rate (also known as hit rate and recall) of a classifier is estimated:

\[
\text{tp rate} \approx \frac{\text{Positives correctly classified}}{\text{Total positives}}
\]

The false positive rate or the false alarm rate is given by:

\[
\text{fp rate} \approx \frac{\text{negatives incorrectly classified}}{\text{total negatives}}
\]

From the ROC curves,

Sensitivity = Recall

\[
\text{specificity} = \frac{TN}{FP + TN}
\]

\[= 1 - \text{fp rate}\]

Positive predictive value = precision
The classifier performance is depicted in two-dimension by the ROC curve. For comparison of classifiers, the ROC performance must be represented in a single scalar vector of expected performance. The commonly used technique is to calculate the area under the ROC curve [61]. The area under the curve of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. Thus, classifiers with greater area are better average performance.

4.1.3 WEKA

WEKA stands for Waikato Environment for Knowledge Analysis. Weka is a machine learning software written in Java. It supports uniform interface to various learning algorithms and also for several data mining process such as preprocessing, clustering, classification and so on. The learning methods are applied on dataset using Weka and the output is analyzed to extract information about the data. Weka is also used for comparing several learners with respect to the performance in order to choose the best classifier. The performances of classifiers are evaluated using a common module.

Weka allows the user to create, run, modify and analyze experiments. The Weka can be run from the command line or from the interface. The platform offers option of cross-validation, splitting the dataset into train and test sets. The user can choose between
classification and regression depending on the classifiers and the
dataset used. The parameters of the experiment can be changed. The
results obtained can be analyzed using an experiment analyzer.

4.1.4 KC1 Dataset

KC1 dataset is a NASA Metrics Data Program, it is publicly
available for verifying, refuting and improving predictive models of
software engineering. KC1 is a C++ system implementing storage
management for receiving and processing ground data. The dataset
consists of the McCabe and Halstead features extractors of the code.
The measures are module based.

The defect detectors are assessed as follows:

\( a \) = Classifier predicts no defects and module actually has no error.

\( b \) = Classifier predicts no defects and module actually has error.

\( c \) = Classifier predicts some defects and module actually has no error.

\( d \) = Classifier predicts some defects and module actually has error.

The accuracy, probability of detection (pd) or recall, probability of false alarm (pf), precision (prec) and effort is calculated as

\[
Accuracy = \frac{(a + d)}{(a + b + c + d)}
\]

\[
recall = \frac{d}{(b + d)}
\]
\[ pf = \frac{c}{(a + c)} \]

\[ prec = \frac{d}{(c + d)} \]

\[ effort = \frac{(c.\text{LOC} + d.\text{LOC})}{(\text{TotalLOC})} \]

The probability of detection is proportional to the effort; thus, higher rate of detection, more effort is required. Probability of false alarm decreases with increase in detection. This linkage can be observed in receiver operating curve (ROC).

The KC1 dataset consists of 2109 instances and 22 different attributes. The attributes are 5 different LOC, 3 McCabe metrics, 12 Halstead metrics, a branch count and 1 goal-field. The attribute information in the dataset is as follows:

1. LOC : numeric % McCabe's line count of code
2. v(g) : numeric % McCabe "cyclomatic complexity"
3. ev(g) : numeric % McCabe "essential complexity"
4. iv(g) : numeric % McCabe "design complexity"
5. n : numeric % Halstead total operators + operands
6. v : numeric % Halstead "volume"
7. l : numeric % Halstead "program length"
8. d : numeric % Halstead "difficulty"
9. i : numeric % Halstead "intelligence"
10. e : numeric % Halstead "effort"
11. b : numeric % Halstead
12. t : numeric % Halstead’s time estimator
13. IOCode : numeric % Halstead’s line count
14. IOComment: numeric % Halstead’s count of lines of
    comments
15. IOBlank: numeric % Halstead’s count of blank lines
16. IOCodeAndComment: numeric
17. uniq_Op : numeric % unique operators
18. uniq_Opnd : numeric % unique operands
19. total_Op : numeric % total operators
20. total_Opnd : numeric % total operands
21. branchCount : numeric % of the flow graph
22. class: {false,true}% module has/has not one or
    more
Examples from dataset:

Example 1 - 1.1, 1.4, 1.4, 1.4, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 2,2, 2,2, 1.2, 1.2, 1.2, 1.2, 1.4, false

Example 2 - 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, true

Example 3 - 83, 11, 11, 171, 927.89, 0.04, 23.04, 40.27, 21378.61, 0.31, 1187.7, 65, 10, 6, 0, 18, 25, 107, 64, 21, true

4.1.5 Methodology

In this work, data mining techniques and their application for classifying modules [20] which require optimization are proposed to provide guidelines on software reliability. This work is carried out using available NASA dataset, in particular the KC1 dataset for classification and reliability prediction [21]. The classification algorithms used in this study are Random tree, Bayesian Logistic Regression and CART and those are used to classify the software modules using software complexity measures such as LOC measure, Cyclomatic complexity, Base Halstead measures and Derived Halstead measures of the KC1 (NASA) dataset which is available in the public domain.

Weka is a machine learning software written in Java. It supports several data mining process such as preprocessing,
clustering, classification and so on. All classification in this study is carried out on Weka.

In this research, probability based classifiers, decision tree based classifiers and neural networks are used. The classification algorithms used are

- Probability based classifiers
- Decision tree based classifiers

For the experiments 2107 samples from the KC1 Dataset are used, using 1391 samples as training set and 716 samples for testing.

4.1.6 Result and Discussion

Weka was used on KC1 dataset for classification and the result is summarized in Table 4.1 and Figure 4.1 and Figure 4.2.

Table 4.1. Classification performance of Random tree, CART and BL regression on KC1 dataset

<table>
<thead>
<tr>
<th>KC1 Dataset</th>
<th>Correctly classified %</th>
<th>Root mean squared error</th>
<th>Mean Absolute error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random tree</td>
<td>81.86</td>
<td>0.43</td>
<td>0.1924</td>
</tr>
<tr>
<td>CART</td>
<td>84.91</td>
<td>0.35</td>
<td>0.2095</td>
</tr>
<tr>
<td>Bayesian logistic regression</td>
<td>86.03</td>
<td>0.37</td>
<td>0.1397</td>
</tr>
</tbody>
</table>
It is observed that Bayes Logistic Regression (BLR) provides the best classification accuracy. However from Figure 4.3 it is observed that the difference between Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) is minimum for CART which indicates
the variance of the error. The confusion matrix obtained for the three classifiers is shown in Table 4.2.

Table 4.2: The confusion matrix for Random tree, CART and BL regression on KC1 dataset

<table>
<thead>
<tr>
<th></th>
<th>Random Tree</th>
<th>CART</th>
<th>Bayesian Logistic Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No defect</td>
<td>Defect</td>
<td>No defect</td>
</tr>
<tr>
<td>No defect</td>
<td>559</td>
<td>58</td>
<td>592</td>
</tr>
<tr>
<td>Defect</td>
<td>72</td>
<td>27</td>
<td>83</td>
</tr>
</tbody>
</table>

The sensitivity and specificity plotted based on equation (4.1) and (4.2) is shown in Table 4.3 and Figure 4.3.

\[
sensitivity = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}} \quad (4.1)
\]

\[
specificity = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}} \quad (4.2)
\]

Table 4.3: Sensitivity and Specificity for the classifiers Random tree, CART and BL regression

<table>
<thead>
<tr>
<th></th>
<th>Sensitivity</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random tree</td>
<td>0.91</td>
<td>0.32</td>
</tr>
<tr>
<td>CART</td>
<td>0.88</td>
<td>0.39</td>
</tr>
<tr>
<td>Bayesian logistic regression</td>
<td>0.86</td>
<td>0.43</td>
</tr>
</tbody>
</table>
From Table 4.3 it is observed that the classifiers perform extremely well with true positives which could be attributed to the higher class labels available for training.

![Sensitivity and Specificity Chart](image)

**Figure 4.3.** The sensitivity and specificity measured

The ROC characteristics of the three classifiers are shown in Figures 4.4, 4.5 and 4.6.

![ROC Curve](image)

**Fig. 4.4:** Receiver operating characteristic of Random Tree
Fig. 4.5: ROC of classification and regression tree

ROC curve shows the tradeoff between sensitivity and specificity. ROC curves are plotted with the false positive rate as x axis and true positive rate as y axis.

Fig. 4.6: ROC of Bayesian Logistic Regression
Figures 4.4, 4.5 and 4.6 show the area under curve with very low values which shows the classifiers as inefficient for predicting faults when used as such. Of the three classifiers CART shows the best values of Area Under Curve (AUC).

4.2 Severity Based Code Optimization: A Data Mining Approach

Severity of errors in a module can be classified as:

1. **Catastrophic**: Defects that could (or did) cause devastating consequences for the system.

2. **Severe**: Defects that could (or did) cause very severe consequences for the system.

3. **Major**: Defects that could (or did) cause significant consequences for the system in question – there is a workaround only after fixing a defect.

4. **Minor**: Defects that could (or did) cause small or insignificant consequences for the system in question. Easy to recover or work around.

5. **No Effect**: Trivial defects that can cause no negative consequences for the system in question. Such defects normally produce no erroneous outputs.
4.2.1 Introduction

Legacy systems made of billions of lines of code are currently running. Majority of legacy system are running critical systems. Both large and small organizations extensively rely on IT infrastructure as the backbone. One of the top priorities of any IT manager is to maintain the existing legacy system and to optimize modules where required to meet current demanding requirements. In order to assess the severity of a software problem, various techniques have been developed to determine the complexity of the modules as well as protocols which are developed.

In this work it is proposed to investigate data normalization algorithms on the KC1 NASA dataset and compute the classification accuracy on a multi class problem. The pre processed data is classified using CART, Bayesian Logistic Regression and Random Tree.

4.2.2 Methodology

Converting data into structures which are appropriate for the data mining algorithm improves the performance efficiency of the data mining algorithm. Normalization techniques range the attribute data into a small specified range, such as -1.0 to 1.0, or 0 to 1.0.
The proposed method of data preprocessing depends on the class attribute of the training dataset. For non defective modules the attributes are normalized using the normal distribution function and for defective modules, the attributes are normalized using the cumulative distribution function.

For a given attribute \( X \) with values \((x_1,x_2, \ldots x_n)\) the normalized values are given by

\[
X[n] = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x - \mu)^2}{2\sigma^2}} x[n]
\]

When no defects are present in the module

\[
X[n] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x - \mu)^2}{2\sigma^2}} x[n] dx
\]

When the module contains a defect.

Where \( \mu \) is given by

\[
\mu = \frac{1}{n} \sum_{i=1}^{n} x_i
\]

and \( \sigma \) is given by

\[
\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2}
\]
4.2.3 Results and Discussion

The classification accuracy of the three classifiers used in the previous section is shown in Table 4.4 Figure 4.7.

Table 4.4: Classification accuracy for the classifiers Random tree, CART and BL regression

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Correctly classified %</th>
<th>Root mean squared error</th>
<th>Mean Absolute error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random tree</td>
<td>94.55</td>
<td>0.43</td>
<td>0.1924</td>
</tr>
<tr>
<td>CART</td>
<td>95.67</td>
<td>0.35</td>
<td>0.2095</td>
</tr>
<tr>
<td>Bayesian logistic regression</td>
<td>96.79</td>
<td>0.37</td>
<td>0.1397</td>
</tr>
</tbody>
</table>

Fig 4.7: Classification accuracy using the proposed normalization technique.

Compared to the results in the previous section it is seen that the proposed method of data normalization improves the classification accuracy.
The sensitivity and specificity of the proposed method for all the three classifiers is shown in Figure 4.8.

![Figure 4.8: Sensitivity and specificity](image)

Though the classification accuracy of Bayesian logistic regression is higher than CART, the variance of error is lower in CART compared to Bayesian Logistic Regression.

The comparative chart between the classification accuracy before and after normalization is shown in Figure 4.9. It is observed that the classification accuracy improves on an average of 10% when preprocessed with the proposed methodology.
Fig 4.9: Comparison of classification accuracy

The ROC curve of Random tree, CART and Bayesian logistic regression after the proposed normalization is shown in Figures 4.10, 4.11 & 4.12.

Fig. 4.10: ROC of Random Tree
Fig. 4.11: ROC of Classification and Regression Tree.

Fig. 4.12: ROC of Bayesian logistic regression
From Figures 4.10, 4.11 & 4.12 it is seen that the area under curve having values of greater than 0.9 which is highly desirable for accurate defect prediction.

### 4.3 Conclusion

In this chapter investigations were carried out to find the efficiency of existing classifiers for software defect prediction. Though the classification accuracy obtained is greater than 80%, it is found that the area under curve does not exceed 0.6 and hence the classifiers are not suitable for real time deployment. To overcome these issues, a novel pre processing technique was proposed. The classification accuracy improved by an average of 10% with the area under curve of the ROC achieving more than 0.9. The proposed technique can be used to improve the classification algorithm. In the next chapter it is proposed to implement novel classifier to further improve the classification accuracy.

The obtained result is compared with other methods available in literature is shown in Table 4.5 and Figure 4.13. It is seen that the proposed technique shows good classification accuracy compared with other techniques in literature except for Lessmann et al., 2008.
Table 4.5: Comparison of classification accuracy with other techniques in literature

<table>
<thead>
<tr>
<th>Technique</th>
<th>% accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rodriguez et al., 2007</td>
<td>92</td>
</tr>
<tr>
<td>Chitra et al., 2010</td>
<td>82.5</td>
</tr>
<tr>
<td>Lessmann et al., 2008</td>
<td>97</td>
</tr>
<tr>
<td>Proposed Normalization technique</td>
<td>96.79</td>
</tr>
</tbody>
</table>

Figure 4.13: Classification accuracy compared with other techniques found in literature