Analysis of Software Fault Prediction Metrics

Rajni Sehgal and Deepti Mehrotra
AUUP, India
Director ASCS, Noida, India

Abstract: This research emphasizes the need to find out the faults from software system to evaluate it for its quality. Software metrics can be used to collect information regarding structural properties of a software design which can be further statistically analyzed, interpreted and linked to its quality. Metrics information can be collected from internal design like logic & algorithmic details as well as from the external design i.e. interfaces of software. There are various software metrics exists which can measure the faults from the software system, in this paper various faults prediction and classification techniques has been studied to find out the faults from the software system. This research identifies a set of metrics to measure and classify the faults from the software systems.

Key words: Metrics, Faults, Error

INTRODUCTION

To improve the quality of a system there is a need to find out the faults from the system. There could be many reasons for system to be faulty, most of the faults are due to the human factor; mistakes and errors made in designing or coding by people, errors made by a software team during specification, design, coding, data entry and documentation, communication failure. To identify the faults we need metrics which can measure the faults from the system there are various metrics which can measure the faults at various phases of software life cycle. Software metrics help to measure structural properties of an artifact. There is need to define metrics based on the formal specifications so that they can be Theoretically as well as empirically Validated. Faults can be easily recognized by using the metrics, which may lead to good quality software.

Related Works: At different levels of software life cycle, software exists in different forms. It may be a software Requirements specification document in the analysis phase, a design pattern in the design phase, or an executable Software. Cartwright, M. Shepperd, M found that there was a significant difference in the defect densities between those classes that participated in inheritance structures and those that did not [1]. Xiang-Sun Zhang found that Instead of performing a program consisting of instructions sequentially as in a von Neumann computer, artificial neural nets have their structures in dense interconnection of simple computational elements [2] Norman E. Fenton review literature to find out the wide range of prediction models use size and complexity metrics to predict defects [3]. Hall T, Fenton N identify consensus requirements for metric program success and examine how programs in two organizations measured up [4]. Basili V.R. et al. presents the results of a study in which they empirically investigated the suite of object-oriented (00) design metrics [5]. J.C. Munson et al., uses the statistical technique of discriminate Analysis as a tool for the detection of fault-prone programs is explored [6] Porter. A et al. Proposed an approach that derives models of problematic components, based on their measurable attributes and those of their development processes [7]. V. Basili and H.D. Rombach purposes an improvement-oriented software engineering process model was developed that uses the goal/question/metric paradigm to integrate the constructive and analytic aspects of software development [8]. S. Henry and D. Kafura purposes a metrics, based on the number of possible paths of information flow through a given component, were used to evaluate the design and implementation of a software system (the UNIX operating
Table 1: Fault Prediction Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>SDLC phase</th>
<th>Benefits</th>
<th>Drawback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of Code)</td>
<td>Predicted the faults in Early life cycle of software Development</td>
<td>This metric calculate the faults by</td>
<td>It depends on programming language used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The total number of lines</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The number of blank lines in module</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The number of lines of comments in a module</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lines of executable code</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The number of lines which contain both code and comment in a module</td>
<td></td>
</tr>
<tr>
<td>Cyclomatic complexity</td>
<td>Faults can be found after the development of the software i.e. later stage of SDLC</td>
<td>The complexity of software can be correlated with the complexity of the graph.</td>
<td>On changing the number of lines of code number of paths also changed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>McCabe proposed the cyclomatic number V(G) which is equal to the number of linearly independent paths through a program in its graphs representation to indicate the software complexity.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The V(G) for a program control graph G, is given by:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ V(G) = E - N + P ]</td>
<td></td>
</tr>
<tr>
<td>Design Complexity</td>
<td></td>
<td>Design complexity measures the amount of interaction between the modules in a system.</td>
<td></td>
</tr>
<tr>
<td>Essential Complexity</td>
<td></td>
<td>Essential Complexity (eV (G)) is a measure of the degree to which a module contains unstructured constructs.</td>
<td></td>
</tr>
<tr>
<td>Halstead Metrics</td>
<td>Predict the faults before the faults propagates to testing phase</td>
<td>Halstead metrics are computed statically from the code and was introduced by Halstead in 1977[11] Metrics applicable to several aspects of program. The metrics are defined as follows. The following token counts are used to compute the various Halstead metrics The metrics are defined as follows. The following token counts are used to compute the various Halstead metrics</td>
<td>It depends upon the tokens of the programming language</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ n_1 = \text{the number of distinct operators} ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ n_2 = \text{the number of distinct operands} ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ N_1 = \text{the total number of operators} ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ N_2 = \text{the total number of operands} ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Halstead length content</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ N = N_1 + N_2 ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Halstead volume metric</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Volume metric is a measure of the storage volume required to represent the program.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ V = N \cdot \log_{2,n} \text{where } n = n_1 + n_2 ]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of Faults</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ \text{Faults} = V/S_0 ]</td>
<td></td>
</tr>
</tbody>
</table>

Halstead et al., purposed the metrics to predict the fault in early life cycle of software development model [11]. Zhou Jianhong et al., explored five Neural Network Based techniques and comparative analysis is performed for the modeling of severity of faults present in function based software systems [12]. Yuan, Dongliang et al., presents a method to evaluate the software reliability
using Fuzzy-Neural network. They established a reliability prediction model based on adaptive-network based fuzzy inference system (ANFIS) [13]. Bezerra, M.E.R et al., introduces a novel algorithm based on constructive RBF neural networks aimed at predicting the probability of errors in fault-prone modules; it is called RBF-DDA with Probabilistic Outputs and is an extension of RBF-DDA neural networks [14]. Bradley valuate six machine learning algorithms (C4.5, Multiscale Classifier, Perceptron, Multi-layer Perceptron, k-Nearest Neighbours and a Quadratic Discriminant Function) on six “real world” medical diagnostics data sets [15]. Yi (Cathy) Liu et al., Purposes genetic-programming-based approach includes three strategies for modeling with multiple software projects: Baseline Classifier, Validation Classifier and Validation-and-Voting Classifier [16].

**Data Mining Techniques of Defect Prediction:** An emerging approach for defect prediction is the use of data mining techniques to predict the problematic areas in the software.

**Naïve Bayes Classifier:** A naïve Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with strong (naïve) independence assumptions. The underlying probability model is an independent feature model. Mathematically classifiers map a discrete or continuous feature space $X$ to a discrete set of labels $Y$. The Naïve Bayes classifiers are efficiently trained in supervised learning due to the precise nature of the probability model. Maximum likelihood methodology is used to find the parameter estimates in the Naïve Bayes models. The advantage of Bayesian models is the easy incorporation of different significant but non-quantifiable factors, such as quality of verification and validation activities, software complexity and test coverage in the model. Mingxi et al., utilized a modified approach effectively to solve the error classification problem [7].

**Bayesian Logistic Regression:** In a supervised learning problem, if an unknown target function $f: X \rightarrow Y$ or equivalently $P(Y|X)$ is to be approximated, assume $Y$ is a Boolean-valued random variable and $X$ is a vector containing $n$ Boolean attributes. In other words, $X = (X_1, X_2, \ldots, X_n)$, where $X_i$ is the Boolean random variable denoting the $i^{th}$ attribute of $X$.

Applying Bayes rule, it is learnt that $P(Y = y_i|X)$ can be represented as:

$$P(Y = y_i|X) = \prod_{i=1}^{n} P(X_i = 1|X)$$

where $y_i$ denotes the $k^{th}$ possible vector value for $X$.

**Decision Tree Induction:** The structure of a decision tree looks like flow-chart; internal nodes representing a test on an attribute, branches symbolize outcome of the test and the leaf nodes denoting classes or class distributions. The root node forms the topmost node in a tree.

Unknown sample is classified by testing the sample against the decision tree, using values of the attributes. A traceable pathway from the root to a leaf node holds the class prediction for that sample. The classification rules used are easily got from decision trees.

**Tree Pruning:** Many of the branches of a decision tree when built will reflect irregularities in the training data. The irregularities occur due to noise or outliers. Tree pruning methods discovers this problem of over fitting the data. Pruning methods normally use statistical procedures to eliminate the least reliable branches. This elimination usually results in quicker classification and an enhancement in the capability of the tree to appropriately classify independent test data. In the pre pruning method, a tree is “pruned” by stopping its construction early; thus no further split or partition at a given node. On halting, the node becomes a leaf which may now hold the most recurrent class in the subset samples, or the probability distribution of those samples.

**Random Tree:** Random tree is a decision tree that considers $K$ randomly chosen attributes at each node and allows class probabilities based on backfitting with no pruning (Frederick Livingston, 2005) [13]. The effects of different variables are generally not found. The steps involved in a random tree are $A$ data set [inbag] is created from the training set by sampling with replacement members. The number of examples in the [inbag] data set is equal to that of the training data set. This new data set may contain duplicate examples from the training set. Using the bootstrapping technique, usually one third of the training set data is not present in the inbag. This left over data is known as the out-of-bag data. This above process is called bootstrapping each tree is made up of a random number of attributes. Nodes and leaves are formed using the attributes with standard tree building algorithms. Pruning is not done and tree is grown to maximum extent possible. From Figure 1 it is seen that the split of the nodes can occur using different logic for the given class labels. Taking the medial value of all the six different trees produces better results.
Fig. 1: Six different trees produced within random tree

Fig. 2: Classification and regression tree

**Classification and Regression Tree (CART):**
Classification and regression tree (CART) is a non-parametric technique [5] that produces either classification for categorical variable or regression trees for numeric variable. Trees formed depend upon the values of variable in the modeling dataset from which a collection of rules is created. Rules are chosen based on the capability of splits formed on variables' values can differentiate observations based on the dependent variable. Child node formed by splitting a node into two, the rule is applied in the case of parent node. CART stops splitting when it detects that there is no further splitting. Terminal nodes at end of each branch.

Each observation falls into one and exactly one terminal node Set of rules define each terminal node uniquely Figure 2 shows the functioning of CART.

From Figure 2 it is seen that each attribute is taken as a node and a binary split is made on some conditions. The condition for split can be based on statistical value or based on other parameters like Gini index. The nodes terminate with the required class label. In the figure there a total of 6 attributes and 4 classes.

**Neural Network:** Artificial Neural Networks are a programming model that seeks to imitate the computational analogues of neurons and are used widely in artificial intelligence problems from simple pattern detection activities to advanced symbolic operations. The Multilayer Perception is an example of an artificial neural network that is used extensively for the solution of a number of different problems, including pattern recognition and interpolation. Artificial Neural Networks endeavors to model the working of the human brain. The human brains consist of millions of individual cells which are called neuron. All experience and knowledge is encoded by the links that are present between neurons.
Given that the human brain consists of such a large number of neurons, the quantity and nature of the connections between neurons is, at present levels of understanding, almost impossible to assess. The network in Figure 3 has an input layer with three neurons, one hidden layer with three neurons and an output layer with three neurons. Each neuron in the input layer represents one predictor variable, but in the definite variables, A-1 neurons are used to represent the A number of the variable.

**Fuzzy Logic:** Logic deals with true and false. A proposition can be right on one instance and false on another. For example, “Rose is red”, if the rose held is yellow, the proposition that rose is a red flower is false or if the rose is of a red variety, then proposition is true. If a proposition is true, it has a truth value of 1; if it is false, its truth value is 0. 0 and 1 are the only possible truth values. Using logical operations, propositions can be combined to generate other propositions.

**Analysis of Software Defect Prediction:** 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 care 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. 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.

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.

**Receiver Operating Characteristics (ROC):** Receiver Operating Characteristics (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 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:

\[
|\text{tp rate} &= \frac{\text{positives correctly classified}}{\text{total positives}} |
\]

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

\[
|\text{fp rate} &= \frac{\text{negatives incorrectly identified}}{\text{total negatives}} |
\]

From the ROC curves,

\[
\begin{align*}
\text{Sensitivity} &= \text{Recall} \\
\text{specificity} &= \frac{TN}{FP + TN} \\
&= 1 - \text{fp rate} \\
\text{Positive predictive value} &= \text{precision}
\end{align*}
\]

**RESULTS AND DISCUSSION**

It is observe that Bayes Logistic Regression (BLR) provides the best classification accuracy. The confusion matrix obtained for the three Classifiers is shown in Table 4.

The sensitivity and specificity plotted based on equation (2.8.1) and (2.8.1) is shown in Table 5 and Figure.

\[
\text{Sensitivity} = \frac{\text{number of true positives}}{\text{no. of true positives + no false negatives}} \tag{2.8.1}
\]

\[
\text{Specificity} = \frac{\text{no. of true negatives}}{\text{no. of true negatives + no flase positives}} \tag{2.8.2}
\]

From Table 5 it is observed that the classifiers perform extremely well with true positives which could be attributed to the higher class labels available for training.

The ROC characteristics of the three classifiers are shown in Figures 7 and 8. 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.

Figures 6, 7 and 8 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).

**Severity Based Code Optimization: A Data Mining Approach:** Severity of errors in a module can be classified as:

- Catastrophic: Defects that could (or did) cause devastating consequences for the system.
- Severe: Defects that could (or did) cause very severe consequences for the system.
- Major: Defects that could (or did) cause significant consequences for the system in question – there is a work around only after fixing a defect.
- Minor: Defects that could (or did) cause small or insignificant consequences for the system in question. Easy to recover or work around.
- No Effect: Trivial defects that can cause no negative consequences for the system in question. Such defects normally produce no erroneous outputs.

The proposed method of data preprocessing depends on the class attribute of the training dataset. The attributes are normalized using the cumulative distribution function. For a given attribute X with values (x1,x2, . . . xn) 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

\[
X[n] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x - \mu)^2}{2\sigma^2}} x[n] dx
\]
Table 2: Tools used during the study

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEKA</td>
<td>It supports uniform interface to various learning algorithms and also for several data mining process such as preprocessing, clustering, classification and so on.</td>
</tr>
<tr>
<td>Data Set</td>
<td>KC1 dataset is a NASA Metrics Data Program taken for this study[17]</td>
</tr>
</tbody>
</table>

Table 3: 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>

Table 4: 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>No defect</td>
<td>559</td>
<td>592</td>
<td>613</td>
</tr>
<tr>
<td>Defect</td>
<td>72</td>
<td>83</td>
<td>96</td>
</tr>
</tbody>
</table>

Table 5: Sensitivity and Specificity for the classifiers Random tree, CART and BL regression

<table>
<thead>
<tr>
<th>Classifier</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>

Fig. 4: Classification accuracy on KC1 dataset

Fig. 5: The RMSE and Mean absolute error
Fig. 6: The sensitivity and specificity measured

Fig. 7: Receiver operating characteristic of Random Tree

Fig. 8: ROC of classification and regression tree
When the module contains a defect. where $\mu$ is given by

$$\mu = \sum_{n_{ij}}$$
The classification accuracy of the three classifiers used in the previous section is shown in Table 5 Figure 9. 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 11. 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.

CONCLUSION

In this study various software defects and their corresponding metrics were studied. Defect prediction techniques using data mining algorithms available in literature were studied. In the next chapter it is proposed to study the KC1 Dataset and existing data mining algorithms to review their classification accuracy. In this Paper 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.

REFERENCES


17. http://promise.site.uottawa.ca/.