Object Oriented Design Metrics for Predicting Fault Proneness using Naïve Bayes and Random Forest

Vaishnavi.J1, Anousouya Devi.M2 and Albert Alexe.A3
1 Assistant Professor, Department of Information and Technology
Dr. SJS Pauls Memorial College of Engineering & Technology, Puducherry, India.
   Email: roshugee@gmail.com
2,3 Assistant Professor, Department of Computer Science & Engineering
Achariya College of Engineering Technology,
Puducherry, India.
   Email: anousouyadevi@gmail.com, albertalex@outlook.com

Abstract— Software testing is an essential part of the software development. Reducing the defects of the software is the software development goal, particularly applicable for high assurance software systems. The object oriented technology insertion into the software industry has created new challenges which use software metrics as a tool for controlling Object oriented design metrics is found to be the dominant method for quality prediction of object oriented programs. Among the different object oriented metrics Chidamber and Kemerer’s OO metrics is found to be more useful for the prediction of fault proneness. The relationship between metrics and fault proneness are conducted through machine learning algorithms. The result of our work is based on the naïve bayes and random forest to improve the accuracy of the fault prediction. Naive bayes gives the maximum accuracy and its integration with random forest enhances the prediction accuracy which decreases the software complexity and improves the maintainability .To perform our validation accurately we collected data from the PROMISE repository.

Index Terms— OO metrics, early statistical method, machine learning technique, fault prediction.

I. INTRODUCTION

The growing demand for higher operational efficiency and safety in industrial processes has resulted in huge interest in fault detection techniques. The wide variety of real time systems necessitates the need to ensure a system will not fail in any case and will perform as specified. To achieve this we need to have a software fault prediction technique. A number of techniques have been proposed for fault prediction but none of them has proven to be the generalized technique. A faulty software may increase the complexity and constraints , testing costs, etc. The techniques used for software fault prediction includes statistical techniques, regression models, multivariate method and machine learning algorithm such as Artificial neural network (ANN), random forest (RF), Boosting algorithms such as LogitBoost (LB), AdaBoost (AB), Naïve Bayes (NB), Kstar. The use of machine learning algorithm is proven to be the practical way of solving the problems of predicting failure, fault and defect proneness. The main machine learning applications fault diagnosis to

© Elsevier, 2014
predict whether the system is normal state or in one of several fault states. The object oriented technology
insertion into the software industry has created new challenges which use software metrics as a tool for
controlling, monitoring and maintaining the software. It is necessary to provide dependable guidelines that
one may follow to help ensure good object oriented programming practices and write reliable code.
Object oriented metrics is an aspect to be considered. Metrics to be a set of standard against the measurement of
effectiveness of object -oriented analysis technique in the design of the software.

Software of Chidamber & Kemerer metrics is suitable to predict class fault proneness during early phases of
the life cycle. In this paper, we use 2 machine learning methods naïve bayes, random forest to predict the
fault proneness. It is possible to make prediction by using several metrics as in multivariate logistic regression

II. RELATED WORK

In [1], a validation of chidamber and kemerer’s object oriented metrics studying the number of changes
performed in two systems implemented with Ada. Chidamber and Kemerer appeared to be adequate in
predicting the frequency of changes across classes during the maintenance phase. The probability of fault
detection is taken as a dependent variable for statistical model. It uses least square linear regression.

In [2], shows that metrics predictors of fault prone classes and determine whether they can be used as early
indicators of the quality of the software. In this paper they use CK metrics as the best predictors of fault with
statistical models such as univariate, multivariate and descriptive analysis.

In [3], empirically explores the relationship between the object oriented metrics and the probability of fault
prediction. This paper shows object oriented coupling, cohesion and inheritance measures. The methodology
used here are data distribution and logistic regression.

In [4], describes how the object oriented metrics given by Chidamber and Kemerer are calculated to illustrate
how fault proneness detection of the open source web and e-mail can be carried out. It performs two
statistical approaches such as univariate and multivariate and also employed two machine learning such as
neural networks and decision trees to perform the relationship between fault proneness and OO metrics.

In [5], describes the multifunctional approach to capture the relationship between CK metrics and defect
proneness. It uses CK metrics through multifunctional estimation techniques.

In [6], presents ongoing work on using projection pursuit regression model to predict object oriented software
maintainability. The projection pursuit regression is used to solve non-linear problems in Object oriented
software maintainability prediction studies.

In [7], empirically investigated the analysis of object oriented design metrics for predicting high and low
severity faults.

In this, studies empirically on object oriented design metrics which are found to be useful for predicting the
fault proneness of a class in Object Oriented software systems. It investigates the accuracy especially with a
subset of chidamber and kemerer suite. It performs two statistical models such as univariate and multivariate
logistic regression. It includes machine learning methods with the regard of fault proneness prediction in
terms of faults in which precision, correctness and completeness.

In [8], evaluates different predictive model for the real time software defect datasets. It provides a
combination of IR and Instance based learning along with consistency based subset evaluation technique with
a better relatively better accuracy in prediction. Also showed the size and complexity metrics are not
sufficient attribute for prediction accuracy. It concludes Bayesian belief network is one of the techniques that
need to be explored to capture this casual relationship.

In [9], its access and compares seven machine learning techniques Artificial neural network (ANN), Random
Forest (RF), two Boosting algorithms (logistic boost (LB)), AdaBoost (AB), naïve Bayes (NB), Kstar,
Bagging and Statistical technique (LR).

III. STATISTICAL TECHNIQUES

The statistical method can be used to summarize or describe a collection of data called descriptive statistics.
The data may be modeled in a way that accounts for randomness and uncertainty in the observation.

A. logistic regression

Logistic regression is a standard statistical model in which the dependent variable can take either zero or one.
It is suitable for classes under consideration are divided into categories of faulty and non faulty classes for
building software quality classification models.
B. multivariate logistic regressions

The multivariate logistic regression equation is as follows [1]

\[
\text{prob}(X_1, X_2, \ldots, X_n) = \frac{\exp(A_0 + A_1 X_1 + \ldots + A_n X_n)}{1 + \exp(A_0 + A_1 X_1 + \ldots + A_n X_n)}
\]

Here P is the Probability of errors found in a class and X is the metrics of the class. If the value of metrics make P greater than 0.5 the class is found to have fault data. Multivariate logistic regression is used to find the effectiveness of the metrics.

C. ordinary least squares

This method is to estimate the unknown parameters in a linear regression model. This method minimizes the sum of the squared vertical distances between the observed responses in the dataset, and the responses predicted by the linear approximation. The resulting estimator can be expressed by a simple formula given below [2].

\[ y_i = \beta x_i + \varepsilon \]

IV. OBJECT ORIENTED SOFTWARE METRICS

The object oriented software metrics used here are Chidamber and Kemerer(CK), Brito e Abrew metrics(MOOD metrics), Bansiya and Davis (QMOOD) metrics, Robert C. Martin’s metrics and McCabe metric suite.

A. Chidamber and Kemerer metric suite

Classic set of metrics are proposed by Chidamber and kemerer in 1991 for OO software [3] and upgraded in 1994[4]. Chidamber and Kemerer used six metrics for predicting fault proneness such as Weighted methods per class(WMC), Depth of inheritance Tree (DIT),Number of children(NOC),Coupling between objects(CBO), Response of a class(RFC) and Lack of cohesion of metrics(LCOM).It is certified by the researchers as per[5,6,8].

a). Weighted Methods per Class(WMC)

WMC is defined as the weighted sum of all the classes methods [8].It is the sum of complexities of local methods of a class. A class with more member functions are more fault prone .For simple WMC, when all complexities are unity, same as number of class methods.

b). Depth of Inheritance (DIT)

DIT is the maximum number of edges between a given class and a root class is an inheritance depth (0 for a classes which has no base class). A maximum increase in the methods inherited by a class increases the inheritance depth which also increases the complexity of its behavior.

c). Number of Children (NOC)

NOC = number of immediate sub-classes of a class

NOC equals the number of immediate child classes derived from a base class. NOC measures the breadth of a class hierarchy, where maximum DIT measures the depth. Depth is generally better than breadth, since it promotes reuse of methods through inheritance. NOC and DIT are closely related. Inheritance levels can be added to increase the depth and reduce the breadth.

A high NOC, a large number of child classes, can indicate several things:
- High reuse of base class. Inheritance is a form of reuse.
- Base class may require more testing.
- Improper abstraction of the parent class

Misuse of sub-classing. In such a case, it may be necessary to group related classes and introduce another level of inheritance.

High NOC has been found to indicate fewer faults. This may be due to high reuse, which is desirable.
d). Lack of Cohesion in Methods (LCOM)
A class's lack of cohesion in methods (LCOM) metric counts the sets of methods in a class that are not related through the sharing of some of the class's fields. In some of these pairs both methods access at least one common field of the class, while in other pairs the two methods to not share any common field accesses. The lack of cohesion in methods is then calculated by subtracting from the number of method pairs that don’t share a field access the number of method pairs that do.

B. Brito E abreu’s Metric Model (MOOD)
Mood metrics describes the basic structural mechanism of object oriented concepts such as encapsulation, inheritance, polymorphism and message passing. MOOD metrics used methods and attributes to perform operations and to represent the status of the objects.

a). Attribute Hiding Factor (AHF)
It is the ratio of total number visible attributes in a set of classes and the total number available attributes in the set. Measures visibility for a class definition.

b). Attribute Inheritance Factor (AIF)
The ratio of inherited attributes to the total number of attributes available in a class.

c). Method Hiding Factor (MHF)
It is the number of visible methods is a measure of the class functionality. Increasing the overall functionality will reduce MHF.

d). Method Inheritance Factor (MIF)
The ratio of inherited methods to the total number of methods available in a class.

C. Bansiya and Davis Quality Model for Object Oriented Design Metrics (QMOOD)
In QMOOD metrics, computed in design process in which the design quality attributes are based on ISO standards such as reusability, flexibility, functionality, extendibility and understandability.

Average Number of Ancestors (AMOOD_ANA).
Average of DIT for all classes in the system.

Cohesion Among Methods (QMOOD_CAM).
A measure of cohesion that is based on the method signatures in a class. Included for completeness.

Class Interface Size (QMOOD_CIS).
The count of public methods in a class.

Data Access Metric(QMOOD_DAM). The ratio of private or protected attributes to the total number of attributes declared in a class.

Direct Class Coupling (QMOOD_DCC).
A count of classes that accept instances of a given Class as a parameter plus classes including attributes of the given class’ type

Measure of Aggregation (QMOOD_MOA).
The percentage of data declarations such as integers, real numbers, etc.

Measure of Functional Abstraction (QMOOD_MFA).
The ratio of inherited methods to the total number of methods available in a class.

Number of Methods (QMOOD_NOM).
The number of methods in a class. Same as WMC when weights of the methods in the class equal unity.

D. Robert c. Matrin’s metric suite

a). Efferent Coupling(ce)
This includes inheritance, interface implementation, parameter type, and exceptions, efferent coupling between packages (ce) measures the total number of external classes coupled to classes of a package because of outgoing coupling (ce) is mainly appropriate for object oriented systems.
b). Afferent Coupling (ca)
The afferent coupling metric determines the number of classes and interface from other package that depend on classes in the analyses package. It measures the total number of external classes of a package because of incoming coupling. All classes are counted only once. If the package does not contain any class or external classes do not use any of the package classes then the value of ca is zero.

c). Abstractness (A)
It is a ratio of abstract classes and interfaces in package or to the total number of classes in the evaluate package. This linear unit metric measures how abstract a package is, in that it will calculate a ratio of how many abstract classes and interfaces. The range for this metric is 0 to 1. A=0 indicates a completely concrete package while a=1 indicates completely abstract.

d). Instability (I)
It is the ratio of efferent coupling (Ce) to total coupling (Ce + Ca). Such that I = Ce / (Ce + Ca). This metric is an indicator of the package's resilience to change. Instability metric results are within a range of < 0 ; 1 >. A value of zero indicates a completely stable package and a value of one indicates a completely unstable package.

E. McCabe's Metric Suite
The cyclomatic complexity (McCabe) is used to evaluate the complexity of an algorithm in a method. A method with a low cyclomatic complexity need not inevitably mean that the methods are not complex because it may mean that decisions are postponed through message passing. On account of inheritance, cyclomatic complexity cannot be used to measure the complexity of a class, but cyclomatic complexity of individual methods united with other measures can be used to calculate the complexity of a class.

a). Normalized Distance From Main Sequence (Dn)
Normalized Distance from Main Sequence is the perpendicular distance of a package from the idealized line A + I = 1. This metric is a measure of abstractness and stability. Ideal packages are either absolutely abstract or stable (x=0, y=1) or absolutely concrete and unstable (x=1, y=0). The range for this metric is 0 to 1, with D=0 representing a package that is coincident with the main sequence and D=1 representing a package that is as far away from the main sequence as possible[9].

b). Lines Of Code (LOC)
The LOC of a class is the total number lines in the body of the class and its methods excluding empty and comment lines [10]. LOC Metrics calculates total lines of code (LOC), blank lines of code (BLOC), comment lines of code (CLOC), lines with both code and comments (C&SLOC), logical source lines of code (SLOC-L), McCabe VG complexity (MVG), and number of comment words (CWORDS). Physical executable source lines of code (SLOC-P) are determined by subtracting the number of blank lines and comment lines from the total number of lines of source code.

c). Method Lines Of Code (MLOC)
A Method line of code (MLOC) is the total number of non-blank and non-comment lines present inside a method. Total number of lines of code inside method body includes avg, max, and sum but excludes blank and comment lines.

d). Source Lines Of Code (SLOC)
SLOC is normally used to forecast the amount of effort that will be necessary to build up a program, as well as to assess programming productivity or effort once the software is created. There are two main types in SLOC, physical and logical. Physical SLOC is the number of lines in the source code of the An Empirical Validation of Software Quality Metric Suites on Open Source Software for Fault-Proneness Prediction in Object Oriented Systems program including comment lines and blank lines. If a section consists of more than 25% blank lines then the blank lines that are in excess of 25% are ignored. Logical SLOC tries to measure the number of statements instead of lines in a program’s source code [5].

e). Nested Block Depth (NBD)
Avg, max, and sum are included in the calculation of the depth of nested blocks of code.

V. Used Machine Learning Techniques
In this section the machine learning techniques used for model prediction are described.
A. Random Forests

RF was proposed by Bremen and constructs a forest of multiple trees and each tree depends on the value of a random vector. For each of the tree in the forest, this random vector is sampled with the same distribution and independently. Hence, random forest is a classifier that consists of a number of decision trees. The resultant output class is the mode of classes output by the individual trees [11]. The algorithm for constructing the tree is as follows: For M training sets, N variables in the classifier and the variable n (n << N) where n indicates the number of independent variables that determine the decision at the terminal node of the tree. A bootstrap training sample is selected. The best split is based on these n variables in the training set each tree is allow to grow fully and is not pruned. RFs have the following benefits [10]:

1) RFs are simple.
2) RFs are comparatively robust to outliers and noise.
3) RFs provide give useful internal estimates of error, strength, correlation and variable importance.
4) RFs may produce a highly accurate classifier for various data sets.
5) RFs provide fast learning.
6) RF runs efficiently on large databases.
7) RF can handle thousands of input variables without variable deletion.
8) RF gives estimates of what variables are important in the classification.

B. Naive Bayes

A Naive bayes is the best classification algorithm for predicting fault prone software. It is based can probability theorem and user bayes theorem for classification. It is the best method for text classification. With large dataset it gives more accuracy result. The requirement large number of samples not only brings heavy work for previous manual classification but also puts forward a higher request for storage and computing resources during the computer post processing. In some areas its performance can be compared with other machine learning technique like neural networks and decision trees. A more descriptive term of probability model is independent feature model. Naives bayes assumes that the presence or absence of a particular feature of a class is not related to the presence or absence of any other feature, given the class variable. The bayes naïve classifier selects the most likely classification *ND

- Given the attribute values ... an
- This result in

\[ V_{nb} = \text{argmax}_{v_j \in V} P(v_j) \pi P(a_j / v_j) \]

We generally estimate

\[ P(a_j / v_j) = \frac{n_c + mp}{n + m} \]

where
- \( n \) = the number of training examples for \( v = v_j \)
- \( n_c \) = number of examples for which \( v = v_j \) and \( a = a_j \)
- \( p \) = a priori estimate for \( P(a_j / v_j) \)
- \( m \) = the equivalent sample size

VI. RESULT ANALYSIS

Rapidminer is an open source data mining and machine learning tool. This environment can be used to extract meaning from the dataset. Rapidminer tool does the best among the all the data mining tool in terms of
technology and applicability. In this research work, we need 10 cross fold validation. The dataset taken for experiment does not have missing values and the distribution for label attributes include 21 distributions for 23 attributes. The accuracy of the software using rapidminer with two machine learning algorithms such as naïve bayes and random forest are greater. Table 1 shows the performance evaluation of the machine learning algorithm. To access the influence of the machine to varied machine learning techniques, we used the following evaluators:

Accuracy, Precision, Recall,

- Precision: precision denotes the ratio of number of files detected and having positive bug counts and the number files detected as a positive bug count.
- Recall: It is the ratio of buggy files totally classified and the total number of buggy files

<table>
<thead>
<tr>
<th>Machine learning algorithm</th>
<th>Accuracy %</th>
<th>Precision %</th>
<th>Recall %</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>99.70</td>
<td>99.85</td>
<td>99.85</td>
<td>0.000</td>
</tr>
<tr>
<td>RF</td>
<td>99.85</td>
<td>99.85</td>
<td>100</td>
<td>0.450</td>
</tr>
</tbody>
</table>

Using rapidminer the accuracy of the software has been determined. Thus the naïve bayes and random forest has achieved the maximum accuracy, precision and recall. The ROC curves for naïve bayes and random forest are given below in figure 1 and 2. The ROC curve represents the comparison of two operating characteristics such as false positive rate and true positive rate. The ROC and ROC thresholds has been plotted here:

![Figure 1: AUC for naïve bayes](image1)

![Figure 2: AUC for random forest](image2)
VII. CONCLUSION

In this paper, the different machine learning techniques used for predicting the fault proneness using object oriented metrics have been discussed. The result of this work is based on the integration of naïve bayes algorithm with random forest algorithm. Among all the object oriented metrics Chidamber and Kemerer metrics is well suitable for predicting the fault proneness. The machine learning algorithms such as naïve bayes and random forest is the best predicting techniques to improve the accuracy of prediction. Using the rapidminer , a data mining tool is giving the best accuracy with naïve and random forest. As a future enhancement, the integration of these two algorithms have to be carried out to reduce the error rate.

REFERENCES