
1M.Vaijayanthi and 2M. Nithya,
1,2Assistant Professor, Department of Computer Science and Engineering, SNS College of Technology, Coimbatore, India

Abstract—Software defects commonly known as bugs, present a serious challenge for the software developers to predict the bugs and to enhance the system reliability and dependability. The software defects are usually an incorrect output value, exceptions occurred in the source code, failure due to logical errors or due to any syntax errors. As the size of the programs grows and it may contain large number of methods, so, occurrence of bugs become more common and difficult to fix. It will take time to predict the bugs at the individual methods.

Many techniques have been developed to mainly focus on method-level bug prediction. Several features are commonly used for method level bug prediction. To identify the best set of features it is proposed to use Filter Based Feature Selection (FBFS) using Information Gain. The Information Gain value is calculated for estimating the individual features. Based on the Information Gain values, the relevant features will be extracted for evaluation. In this work, the method-level bug prediction will be carried out using Support Vector Machine (SVM) classifier. Finally, the performance of the bug prediction models will be measured by using Precision, Recall and F-measure values. The volume of predicted bugs can be assessed by using the values of evaluation measures.

Keywords: Bug Prediction, Precision, Recall, F-Measure, Method-Level, Information Gain, Accuracy, SVM Classifier.

I. INTRODUCTION

Bug prediction is considered to be an important field of research in software engineering. Usually software defects, such as, indefinite loop or an incorrect output values are commonly known as bugs. Typically bug prediction plays a major role to predict the bugs in the early stages of software development process. Recently many bug prediction techniques are based on method-level. These techniques include several Machine Learning Classifiers like Naïve Bayesian classifier, Support Vector Machine classifier.

Such techniques rely on some software quality metrics (say., Lines of Code, complexity). The software defects or bugs cannot be directly measured, so certain software quality metrics are collected from standard datasets. The dataset includes source code metrics like Chidamber and Kemerer (CK) metrics, Object Oriented (OO) metrics and Change metrics to predict bugs at the method-level.

To evaluate the performance of the bug prediction model, Machine Learning classifier called Naïve Bayesian classifier is used. The first step in classification process is that to construct a model using training set. Then the classifier will use the model to predict the bugs in future with unknown class values as testing set. The performance of the bug prediction model is estimated using some well known performance evaluation criteria like Precision, Recall and Accuracy values.

The accuracy is the degree to which the algorithm correctly identifies future bugs. Precision is defined as the ratio of the number of modules correctly predicted as defective, to the total number of modules predicted in the set [1]. Recall is defined as the ratio of the number of modules predicted correctly as defective to the total number of defective modules in the set.

II. LITERATURE REVIEW

A. Machine Learning

There are two things that need to be achieved in machine learning process. First, the training needs to be done with known class labels. Second, the trained model needs an efficient algorithm to validate the unknown class labels by means of testing. Supervised learning is common in classification problems where the goal is to have the learner learn a predefined classification [1]. Table 1 shows the general structure of data used in supervised learning. Each instance of data is defined by a set of features and a class.

B. Ten-Fold Cross Validation

Cross-validation involves partitioning a sample of datasets into complementary subsets, performing the analysis on one subset (called the training set), and validating on the other subset (called the validation set or testing set) [2].

The validation process simplifies as:
1. Break data into 10 sets of size n/10.
2. Train on 9 datasets and test on 1 dataset.
3. Repeat 10 times and take a mean accuracy.

<table>
<thead>
<tr>
<th>Table 1: General Structure Of Data Used In Supervised Learning With Known Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>N</td>
</tr>
</tbody>
</table>

III. CLASSIFICATION USING NAÏVE BAYESIAN APPROACH

A. Corpus Collection

To experiment with the machine learning classifier, the data are collected from the standard bug prediction dataset such as Lucene dataset. It consist of the source code metrics, change metrics and bug metrics. The main focus of this paper is to construct a model to predict the method-level rather than at the file-level requires that all metrics are available at method-level.

B. Dataset

The standard bug prediction dataset includes source code metrics like Chidamber and Kemerer (CK) metrics and Object Oriented (OO) metrics. The change metrics includes, the file(s) being affected by the changes commonly known as Revision.
The source code metrics include #methods, #fanin, #fanout, and #attributes. Bug prediction is an important challenge in Software Engineering research. The goal is to build reliable predictors that can indicate in advance about those components of a software system that are more likely to fail. Due to its relevance to software quality, various bug prediction techniques have already been proposed.

Essentially, such techniques rely on different predictors, including source code metrics, change metrics, etc. The main focus is to define the relationships between the defined metrics and the occurrences of bugs [3]. The metrics are already defined in the public dataset to evaluate the bug prediction techniques. This dataset provides the change log approaches and the single-version approaches and hence provide the necessary information for the defect prediction. The original dataset includes the Lucene dataset. The metrics that are available in the dataset are [4] Chidamber and Kemerer (CK) metrics and Object Oriented (OO) metrics. There are 6 CK metrics and 11 OO metrics listed in Table 2. Table 2 shows the metrics included in the original dataset.

Table 2: Chidamber and Kemerer Metrics and Object Oriented Metrics

<table>
<thead>
<tr>
<th>TYPES</th>
<th>METRICS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CK</td>
<td>WMC</td>
<td>Weighted Methods Per Class</td>
</tr>
<tr>
<td>CK</td>
<td>DIT</td>
<td>Depth Of Inheritance Tree</td>
</tr>
<tr>
<td>CK</td>
<td>RFC</td>
<td>Response For Class</td>
</tr>
<tr>
<td>CK</td>
<td>NOC</td>
<td>Number Of Children</td>
</tr>
<tr>
<td>CK</td>
<td>CBO</td>
<td>Coupling Between Objects</td>
</tr>
<tr>
<td>CK</td>
<td>LCOM</td>
<td>Lack Of Cohesion On Methods</td>
</tr>
<tr>
<td>OO</td>
<td>FANIN</td>
<td>Number Of Classes That Reference The Class</td>
</tr>
<tr>
<td>OO</td>
<td>FANOUT</td>
<td>Number Of Classes Referenced By The Class</td>
</tr>
<tr>
<td>OO</td>
<td>NOA</td>
<td>Number Of Attributes</td>
</tr>
<tr>
<td>OO</td>
<td>NOPA</td>
<td>Number Of Public Attributes</td>
</tr>
<tr>
<td>OO</td>
<td>NOPRA</td>
<td>Number Of Private Attributes</td>
</tr>
<tr>
<td>OO</td>
<td>NOAI</td>
<td>Number Of Attributes Inherited</td>
</tr>
<tr>
<td>OO</td>
<td>LOC</td>
<td>Number Of Lines Of Code</td>
</tr>
<tr>
<td>OO</td>
<td>NOM</td>
<td>Number Of Methods</td>
</tr>
<tr>
<td>OO</td>
<td>NOPM</td>
<td>Number Of Public Methods</td>
</tr>
<tr>
<td>OO</td>
<td>NOPRM</td>
<td>Number Of Private Methods</td>
</tr>
<tr>
<td>OO</td>
<td>NOMI</td>
<td>Number Of Methods Inherited</td>
</tr>
</tbody>
</table>

C. Code metrics

There are two traditional suits of code metrics exist:
- CK metrics suite.
- Set of metrics directly calculated at the method level.

E. Naïve Bayes Classifier

A Naive Bayes classifier is a probabilistic classifier based on applying Bayes’ theorem with strong independence assumptions. When represented as a Bayesian network, a Naive Bayes classifier has the structure depicted in [4] Figure 1. It shows the independence assumption among all features in a data instance.

F. Algorithm

Let \( X = \{X_1, \ldots, X_n\} \) be a finite set of observed random variables, called features, where each feature takes values from its domain \( D_i \). The set of all feature sets is denoted by \( \Omega = D_1 \times \ldots \times D_n \). Let \( C \), such that \( C \in \{0, u - 1\} \), be an unobserved random variable denoting the class of a set of features.

A hypothesis \( h : \Omega \rightarrow \{0, \ldots, u - 1\} \), that assigns a class to any given set of variables is defined as a classifier. Each class \( c \) is assigned a discriminant function \( f_c (x) \), \( c = 0, \ldots, u - 1 \) [5]. The classifier selects the class with the maximum discriminant function on a given set of variables.

Thus the bayes theorem can be written as:

\[
P(X) = \frac{P(X/H) P(H)}{P(H)} \quad (1)
\]

The equation 1 shows that, it predicts \( X \) belongs to \( C_i \) iff the probability \( P(C_i/X) \) is the highest among all the \( P(C_i/X) \) for all the \( k \) classes.

G. Performance Evaluation

The classifier is performed by evaluating which set of features yields the best overall classification accuracy and recall, and also by examining the relative contributions of individual features. Table 3 explains the confusion matrix.

Precision

Precision is defined as the ratio of the number of modules correctly predicted as defective, to the total number of modules predicted in the set. Precision is also termed as True Positive which classified as truly predicted as bugs.
**Recall**

This metric indicates the coverage of the accuracy. Recall is defined as the ratio of the number of modules predicted correctly as defective to the total number of defective modules in the set. Recall is also termed as True Negative which classified as not bugs.

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

**IV. CLASSIFICATION USING SUPPORT VECTOR MACHINE CLASSIFIER**

**A. Feature Selection**

In feature selection process, the source code metrics and the change metrics are selected from the whole bug prediction dataset. The features will be taken for classifier training once it is evaluated using K-Fold cross validation process [6]. The cross validation process will separate the training set and the testing set.

**B. Information Gain Calculation**

Once the metrics have been collected from the dataset, the small set of features alone will be selected for the evaluation of bug prediction process [7]. The subset of features will be selected from the whole dataset by calculating the information gain for all the features in the dataset.

The Support Vector Machine classifier will take the testing set and will calculate the accuracy and then will take the testing samples to evaluate the accuracy for unknown labels. It is depicted in Figure 2.

**D. Evaluating the Approaches**

The performance of bug prediction approaches is evaluated with several strategies, each according to a different usage scenario of bug prediction. We evaluate each technique in the context of classification (defective/non-defective) [9]. Prior to model building and classification we labeled each method in our dataset as either as bug-prone or not bug-prone as follows:

\[
\text{Performance evaluation of the features}
\]

Since the choice of features can affect the performance of classifiers, each feature’s discriminative power for performing change classification is compared [8]. This is performed by evaluating which set of features yields the best overall classification accuracy and recall, and also by examining the relative contributions of individual features.

The accuracy of the classifier will be estimated by combining the terms Precision and Recall using the F-measure values.

\[
F - \text{Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

**RESULTS**

The classifier is trained with the standard Lucene bug prediction dataset. The dataset contains the set of attributes, classes, set of data and instances. Based on the class values the testing set will predict the values similar to that of the training
values. The original dataset will contain the information regarding the performance evaluation for the bug predictors. Each and every data will be considered as features in the dataset [11]. Once the data is selected as features, the cross validation will be performed for separating the features as training and testing samples. Then it will feed to the classifier for performance evaluation for calculating the accuracy.

The data are collected from the standard bug prediction dataset. The data will be in the form of excel format (say .xlsx). The original dataset will contain the information regarding the performance evaluation for the bug predictors. Each and every data will be considered as features in the dataset. Once the data is selected as features, the cross validation will be performed for separating the features as training and testing samples [12]. Then it will feed to the classifier for performance evaluation for calculating the accuracy.

By analyzing the results for the Naïve Bayesian classifier, the accuracy is evaluated using cross validation method as 10 fold cross validation. By using this classifier, the accuracy obtained is 96.89%. But the accuracy for the support vector machine classifier is 91.02%

The proposed system proposes the feature selection method as Information Gain calculation. To increase the accuracy, the feature selection has to be carried out. The implementation is currently in progress.

SCREEN SHOTS

![Figure 3: Metrics in the Lucene Dataset](image1)

![Figure 4: Accuracy by Naïve Bayesian Classifier](image2)

<table>
<thead>
<tr>
<th>CLASSIFIER</th>
<th>VALIDATION METHOD (k-fold)</th>
<th>ACCURACY (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAÏVE BAYESIAN</td>
<td>k=2</td>
<td>93.87</td>
</tr>
<tr>
<td></td>
<td>k=5</td>
<td>92.8</td>
</tr>
<tr>
<td></td>
<td>k=10</td>
<td>94.8</td>
</tr>
<tr>
<td>SUPPORT VECTOR MACHINE</td>
<td>k=2</td>
<td>90.02</td>
</tr>
<tr>
<td></td>
<td>k=5</td>
<td>91.72</td>
</tr>
<tr>
<td></td>
<td>k=10</td>
<td>95.02</td>
</tr>
</tbody>
</table>

FUTURE WORK AND CONCLUSION

The accuracy of the classifier is evaluated using the performance evaluation measures. For the metrics dataset the Naïve Bayesian classifier performed well and the accuracy has been evaluated with the cross validation process. Finally the accuracy of the classifier is evaluated using the measures like Precision, Recall and F-measure values. The bug prediction process has been carried out with the standard bug prediction dataset.

Future work will include additional metrics related to standard bug prediction dataset and extend the analysis with the advanced feature selection process.

References


