Investigating Software Detection Methods

Kiran Khatter
School of Engineering & Technology
Ansal University
Gurgaon, India
kirankhatter@gmail.com

Arvind Kalia
Department of Computer Science
Himachal Pradesh University
Shimla, India
arvkalia@gmail.com

Abstract— A defective module not only increases the development time and development cost but also increases the maintenance time and maintenance cost. According to the available literature survey, many systems failed due to schedule and time budget overruns. Therefore a software defect detection technique is needed to identify those software modules that are very likely to include defects and thereby improves the software quality by contributing in the efficient removal software defects. The main objective of the paper is to help software developers in identifying the software defects based on the various software metrics using various classification and machine learning techniques. In this paper, we are performing empirical classification comparison on 5 real world datasets.

Keywords— Software defect detection, Clustering Techniques, Machine Learning Techniques, Software Metrics

I. INTRODUCTION

Computer Software have become an essential part in everyday life and have been used for different applications ranging from business and personal computing applications to real-time applications. The main objective of software development is to develop high quality software with the expected functionalities. In context of software engineering, software quality refers to the satisfaction of functional and non-functional requirements. A functional requirement refers to the operations performed by the software and its components whereas Non-functional requirement refers to the quality attributes such as reliability, maintainability, availability etc. of the system and the degree of satisfaction of non-functional requirements determines the success/failure of software systems [1].

There are also various approaches to achieve these attributes with emphasis to improve the software quality by preventing the introduction of faults. These software defect detection approaches help in finding the defective modules in the software. Testing at the various level of software development process is the traditional way to identify the defects, but when the project size increases in terms of lines of codes (LOCs) and complexity, it becomes difficult and expensive to find the defects with the use of traditional testing techniques. If defects or errors are not identified in the initial phase, these may likely to creep in the later phases such as design and implementation phase of the software development process. Repair of such errors and defects increases the time and cost of software development process [2][3][4]. National Institute of Standards and Technology did study on various projects of U.S. and concluded that software errors cost U.S. economy about $59.5 billion yearly. So testing and debugging are persistent during the software development process. [5]. Thus sooner we detect defected software components; it lowers down the cost & time to develop the software and hence improves the reliability of the software. Thus Defect prediction is imperative to accomplish the software quality.

In order to perform comparison of different software fault detection classifiers, we are using various performance metrics such as probability of detection, accuracy, precision, G-mean, F-measure. This paper also represents the comparison of various different classification models using ROC curves.

This paper is organized as follows: Section 2 explains the related work reported by various researchers for software fault detection. Section 3 describes the data sets used in the study to perform comparison. Section 4 specifies the different classifiers and a detailed comparison between various classifiers. Section 5 represents the experimental set up and the results interpretation followed by conclusion of the study.
II. RELATED WORKS

A software defect is an error, failure or fault which triggers software to diverge from its specified behavior. The main objective of software detection method is to categorize the fault prone and non-fault prone modules on the basis of classifier. It improves the overall quality of software by discovering the software defects in the early phase of software development process and helping developers to focus on defect prone modules [3][6][7]. Identification and removal of software faults is one of the critical activities in the software development process. Though it is not possible to make defect-free software, it is feasible to minimize the software faults and their impact on the software. Anticipation and restoration of software defects not only improves the quality of software, also yields the huge returns.

There are various public repositories such as NASA MDP, PROMISE etc. utilized for building classifiers to identify the fault prone and non-fault prone modules. These repositories specifies the information on various metrics such as LOC, Halstead measure, Mccabe Complexity, branch count etc. There are various classification and machine learning methods such as Associative Classification [8], Logistic Regression [9][10], Fuzzy Subtractive Clustering [11], Support Vector Machine[12] and Artificial Neural Networks [13][14]and Baysian Belief network [15][16]etc. are accessible to build software prediction models.

Fenton et al. [17] used Dynamic Bayesian Network for different software life cycles adopted for several platforms.

Greg Rothermell [18] prioritized the test suits for regression testing by using different types of techniques and checked which technique is best for the prediction of software faults.

Janes et al. [19] used zero-inflated negative binomial regression and Poisson distribution for real-time telecommunication systems.

Catal et al. [20] used a clustering based software fault prediction approach to classify software failures.

Seliya et al. [21] used K-means clustering method to investigate fault prone software modules.

Hall et al. [22] used N-fold cross-validation multiple times to transform the software failure data to the model properly and to represent the system with an acceptable accuracy.

Zhong et al. [23] used K-means and Neural-Gas clustering algorithms to cluster modules to perform software fault prediction.

In this paper, we are using PROMISE Software Engineering Repository [24] to build software defect prediction models based on the Classification Tree, Logistic Regression, Support Vector Machine and Naïve Bayes algorithm. Five following datasets are investigated to perform comparison of different classifiers.

A. JM1 Dataset:

JM1 dataset, written in C language, contains 10,885 modules on software metrics and fault-proneness information of Real-time predictive ground system project.

B. CM1 Dataset

CM1 data set contains 498 modules of NASA spacecraft instrument for prediction of fault-prone modules.

C. PC1 Dataset

The PC1 project, written in 40 KLOC of C, has 1,109 modules on the flight software for earth orbiting satellite application.

D. KC1 Dataset

KC1 project, written in C++, consists of 43 KLOC has 2017 modules to process the data.

E. KC2 Dataset

KC2 project is on storage management system to process data and contains 520 modules of C++ code.

III. CLASSIFICATION TECHNIQUES

A. Classification Tree

Classification Tree helps in categorizing the fault prone and not fault prone modules on the basis of certain software metrics or code attributes derived from datasets. These datasets includes the data of previous development projects in terms of number of faults and characteristics code attributes [25] [26]. Voulgaris et al. [27] used classification tree based on the JM1 and CM1 projects of NASA and the model they built had 79.3% accuracy. The advantage of the classification trees is the accuracy of the prediction model, but it cannot evaluate imbalanced trees. The classification tree for JM1 dataset is shown below in the Figure 1:
B. Logistic Regression

Logistic Regression is a non-linear regression technique to measure the relationship between continuous dependent and independent variables. In this technique, maximum-likelihood ratio is used to measure the significance of the variables. It is used for predicting outcomes of class label based on the specific features of an individual case [28][29].

C. Support Vector Machine (SVM)

VM is a machine learning technique used for performing data classification and regression analysis in varied applications such as fingerprint and handwritten recognition etc. SVM helps in separating the clusters of vectors on the basis of the optimal hyper plane. Elish et al. [29][30] compared the performance of SVM against various statistical and machine learning methods and indicated that SVM yields better results in comparison to other methods in the context of NASA datasets.

D. Naive Bayes

Naive Bayes classifier is the probabilistic classifier based on Bayes theorem which represents the networks of probabilities in order to capture the probabilistic relationships between variables. Bayesian network represents joint probability distribution in the form of a Directed Acyclic Graph network structure \((G)\) for \(V\) variables. The arcs symbolize the Bayesian probabilistic relationships among various nodes representing random variables [2]. It represents the group of conditional probability distributions where each variable \(V_i\) in the graph \(G\) is represented by a conditional distribution given its parent nodes \(Par(V_i)\) [2][31]. It consists of following two components:

a) A directed acyclic graph \(G\) whose vertices correspond to random variables \(V_1, V_2, V_n\).

b) Conditional probability distribution of every variable \(V_i\) given its parents \(Par(V_i)\).

Graph \(G\) is based on the Markov assumption “Every variable \(V_i\) given its parents \(Par(V_i)\) is conditional independent of non-descendants nodes” and it is defined as follows [32]:

\[
Par (V_1, V_2, ..., V_n) = \prod_{i=1}^{n} P(V_i | Par (V_i))
\]

IV. EVALUATION MEASUREMENTS

Binary Classifiers are evaluated on the basis of Confusion Matrix, which helps to classify fault-prone, or not fault-prone modules. Figure 2 represents the information about the actual and predicted classification for any two-class classifiers. In this...
study, if software module is found to be defective, is categorized as “positive” case otherwise modules without defects are termed as “negative” cases [8].

<table>
<thead>
<tr>
<th>Modules with Defect?</th>
<th>Defects Predicted?</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fp</td>
<td>NO</td>
<td>fp</td>
</tr>
<tr>
<td>nfp</td>
<td>YES</td>
<td>nfp</td>
</tr>
</tbody>
</table>

Figure 2: Defect Prediction Confusion Matrix

In order to assess the performance of classification algorithm, several performance metrics are defined below on the basis of total number of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) cases:

**A. Probability of Detection (PD)**

It is the proportion of fault-prone modules predicted by the classification algorithm. It is defined as follows [8]:

\[
PD = \frac{TP}{TP + FN}
\]

**B. Accuracy (ACC)**

It refers to the correctly predicted modules and calculated as follows [5]:

\[
ACC = \frac{TP + TN}{TN + FP + FN + TP}
\]

**C. Precision**

The precision is percentage of correctly predicted fault-prone modules, calculated as follows [8]:

\[
Precision = \frac{TP}{TP + FP}
\]

**D. F-measure**

It is the harmonic mean of recall and precision and calculated as follows [8]:

\[
F\text{-measure} = \frac{(\beta^2 + 1) * \text{precision} * PD}{\beta^2 * \text{precision} + PD}
\]

In F-measure, \(\beta\) refers to any non-negative value and controls the weight assigned to PD and precision. If \(\beta\) takes value 1, then PD and precision are weighted equally [8]. Following mean measure consider “imbalance of data” into account while evaluating the classification method [8]:

\[
G\text{-mean}_1 = \sqrt{PD * \text{precision}}
\]

\[
G\text{-mean}_2 = \sqrt{PD * \text{TNR}}
\]

Both G-mean’s and F-measure result in 0 when all the positive cases are predicted incorrectly.

**V. PERFORMANCE COMPARISON**

A suitable way to compare the performance of various classification algorithms is by using Receiver Operating Characteristic (ROC) curves. The ROC curve is two-dimensional graph in which PF is plotted against PD to indicate the tradeoff between true positive cases and false positive cases. The higher the PD at low PF values, the better the model [33]. The goal here is to investigate which classification method performs better in the prediction of fault-prone modules of these data sets. We are using RF algorithm at three cutoffs: (0.9, 0.1), (0.8, 0.2) and (0.7, 0.3).
TABLE I
EXPERIMENTAL RESULTS ON DIFFERENT METRICS-JM1PROJECT

<table>
<thead>
<tr>
<th>Methods</th>
<th>Probability of Detection</th>
<th>Accuracy</th>
<th>Precision</th>
<th>G-mean$_1$</th>
<th>G-mean$_2$</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Tree</td>
<td>0.605</td>
<td>0.826</td>
<td>0.125</td>
<td>0.275</td>
<td>0.71</td>
<td>0.207</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.107</td>
<td>0.824</td>
<td>0.598</td>
<td>0.253</td>
<td>0.325</td>
<td>0.182</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>0.118</td>
<td>0.835</td>
<td>0.838</td>
<td>0.315</td>
<td>0.343</td>
<td>0.208</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>0.491</td>
<td>0.816</td>
<td>0.204</td>
<td>0.317</td>
<td>0.699</td>
<td>0.289</td>
</tr>
<tr>
<td>Random Forest (Cutoff= (0.9,0.1))</td>
<td>0.969</td>
<td>0.839</td>
<td>0.532</td>
<td><strong>0.718</strong></td>
<td><strong>0.886</strong></td>
<td><strong>0.687</strong></td>
</tr>
<tr>
<td>Random Forest (Cutoff= (0.8,0.2))</td>
<td>0.959</td>
<td>0.973</td>
<td>0.901</td>
<td><strong>0.938</strong></td>
<td><strong>0.968</strong></td>
<td><strong>0.929</strong></td>
</tr>
<tr>
<td>Random Forest (Cutoff= (0.7,0.3))</td>
<td>0.956</td>
<td>0.988</td>
<td>0.977</td>
<td><strong>0.967</strong></td>
<td><strong>0.975</strong></td>
<td><strong>0.967</strong></td>
</tr>
</tbody>
</table>

In Table 1, the highest values in both G – mean and F – measure in Random forests at cutoffs (0.9, 0.1), (0.8,0.2) and (0.7, 0.3) indicate that random forests is best classification method in comparison of other classification techniques. Since JM1 data set suffers from noisy software engineering measurements, we can conclude that random forests performs better in comparison to other classifiers on a large and noisy dataset.

TABLE II
EXPERIMENTAL RESULTS ON DIFFERENT METRICS-CM1PROJECT

<table>
<thead>
<tr>
<th>Methods</th>
<th>Probability of Detection</th>
<th>Accuracy</th>
<th>Precision</th>
<th>G-mean$_1$</th>
<th>G-mean$_2$</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Tree</td>
<td>0.636</td>
<td>0.934</td>
<td>0.483</td>
<td>0.554</td>
<td>0.779</td>
<td><strong>0.549</strong></td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.276</td>
<td>0.934</td>
<td>0.800</td>
<td>0.469</td>
<td>0.524</td>
<td>0.410</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>0.069</td>
<td>0.922</td>
<td>1.000</td>
<td>0.263</td>
<td>0.263</td>
<td>0.129</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>0.282</td>
<td>0.867</td>
<td>0.379</td>
<td>0.327</td>
<td>0.515</td>
<td>0.324</td>
</tr>
<tr>
<td>Random Forest (Cutoff= (0.9,0.1))</td>
<td>1.000</td>
<td>0.922</td>
<td>0.518</td>
<td><strong>0.719</strong></td>
<td><strong>0.851</strong></td>
<td>0.518</td>
</tr>
<tr>
<td>Random Forest (Cutoff= (0.8,0.2))</td>
<td>1.000</td>
<td>0.974</td>
<td>0.723</td>
<td><strong>0.967</strong></td>
<td><strong>0.919</strong></td>
<td><strong>0.722</strong></td>
</tr>
<tr>
<td>Random Forest (Cutoff= (0.7,0.3))</td>
<td>1.000</td>
<td>0.994</td>
<td>0.935</td>
<td><strong>1.000</strong></td>
<td><strong>0.997</strong></td>
<td><strong>0.967</strong></td>
</tr>
</tbody>
</table>

From the outcome in Table 2, Classification Tree appears to perform better on CM1 project based on F – measure. Random forests at cutoffs (0.8, 0.2) and (0.7, 0.3) returns the highest values for both G – mean and F – measure. It demonstrates that Random Forest Algorithm is in better execution.
In Table 3, Random Forest algorithms at cutoffs (0.9, 0.1), (0.8, 0.2) and (0.7, 0.3) and Logistic Regression show better preference in comparison to Classification Tree, SVM and Naïve Bayes methods. Classification Tree produces an acceptable value in G-mean, but a low value in F-measure.

Logistic Regression produces the best result based on G-mean and F-measure. Random Forest Algorithm also gives not too bad performance measures. It appears that aside from Logistic Regression, Random forest algorithm accomplishes higher results at cutoffs (0.9, 0.1), (0.8, 0.2) and (0.7, 0.3).
Table 5 indicates that Classification Tree appears to perform better on KC2 project based on G−mean. Random forests at cutoffs (0.9, 0.1), (0.8, 0.2) and (0.7, 0.3) returns the highest values for G-mean and F−measure.

We have used two-dimensional ROC curve to represent the trade-off between True Positive cases and False Positive cases for assessing the performance of various classification algorithms being used in the study: Classification Tree, Logistic Regression, Support Vector Machine, Naïve Bayes and Random Forest Algorithms.

Table: Experimental Results on Different Metrics: KC2 Project

<table>
<thead>
<tr>
<th>Methods</th>
<th>Probability of Detection</th>
<th>Accuracy</th>
<th>Precision</th>
<th>G-mean_1</th>
<th>G-mean_2</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification Tree</td>
<td>0.825</td>
<td>0.906</td>
<td>0.712</td>
<td>0.767</td>
<td>0.873</td>
<td>0.765</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.466</td>
<td>0.85</td>
<td>0.739</td>
<td>0.587</td>
<td>0.667</td>
<td>0.571</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>0.384</td>
<td>0.85</td>
<td>0.824</td>
<td>0.562</td>
<td>0.612</td>
<td>0.523</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>0.696</td>
<td>0.838</td>
<td>0.438</td>
<td>0.552</td>
<td>0.774</td>
<td>0.538</td>
</tr>
<tr>
<td>Random Forest (Cutoff=(0.9,0.1))</td>
<td>0.932</td>
<td>0.829</td>
<td>0.562</td>
<td>0.734</td>
<td>0.864</td>
<td>0.701</td>
</tr>
<tr>
<td>Random Forest (Cutoff=(0.8,0.2))</td>
<td>0.917</td>
<td>0.718</td>
<td>0.817</td>
<td>0.938</td>
<td>0.931</td>
<td>0.865</td>
</tr>
<tr>
<td>Random Forest (Cutoff=(0.7,0.3))</td>
<td>0.917</td>
<td>0.971</td>
<td>0.866</td>
<td>0.944</td>
<td>0.931</td>
<td>0.931</td>
</tr>
</tbody>
</table>

Figure 3: ROC Curve- JM1 Project
Figure 4: ROC Curve- CM1 Project

Figure 5: ROC Curve- PC1 Project
Figure 6: ROC Curve – KC1 Project

Figure 7: ROC Curve – KC2 Project
Figure 3 represents Random Forest as best classification algorithm in terms of identifying defective software modules. From Figure 4 and Figure 7, it is evident that Classification Tree works in better execution for CM1 and KC2 project. Figure 5 represents Logistic Regression and Random Forest as better classification methods. ROC curve shown in Figure 6 represents the Logistic Regression as best classification algorithm representing the trade-offs between True and False cases. Thus ROC plots shown in Figure 3, Figure 4, Figure 5, Figure 6 and Figure 7 indicates the performance comparison of Classification Tree, Logistic Regression, Support Vector Machine, Naïve Bayes and Random Forest Algorithms. It is found that Random Forest algorithm works well on JM1 project of PROMISE Software Engineering Repository. The performance of Random forest algorithm is good if voting thresholds is proportionate to number of fault-prone modules.

VI. CONCLUSIONS

As software defects are introduced in software development process, modules in the dataset categorized into fault prone and not fault prone modules. The early detection of software defects decreases cost of development and increases the overall quality in software systems. This paper gave several insights in the context of software fault prediction. Different software fault prediction algorithms were evaluated in terms of different prediction performance parameters. It is also reported that performance of classification algorithm may return arbitrary results depending on voting threshold values.

References


© IEOM Society International
Proceedings of the 2016 International Conference on Industrial Engineering and Operations Management  
Kuala Lumpur, Malaysia, March 8-10, 2016


**BIOGRAPHY**

**Dr. Kiran Khatter** is currently working as Assistant Professor in the School of Engineering & Technology at Ansal University, Gurgaon. She is awarded Ph.D. in Computer Science from Himachal Pradesh University, Summer Hill, Shimla. She has done M.Tech (IT) from Punjabi University, Patiala in 2004. She has over 12 years of experience in IT industry and Academics. She has a number of publications in journal & international conference proceedings of repute. Her research interests span over Software Engineering, Image Processing, Data Mining and Big Data analytics. She has keen interest in computer programming. She is an Oracle Certified Java SE 6 Programmer. She has done training in ABAP programming and handled ABAP Projects.

**Dr. Arvind Kalia** is currently working as a Professor at Himachal Pradesh University, Shimla. He has done Master of Computer Applications from Thapar Institute and Ph.D from Punjabi University, Patiala. He has authored two books with National and International Publishers. He has good number of publications in national and international journals of repute. He has an experience of over 25 years and has been interested in the area of Software Engineering and Data mining. Currently, he is involved in research related to Software Quality, Component Based Systems and Open Software Systems. He is also a life member of Computer Society of India and Indian Science Congress.