Verification of the effectiveness of fuzzy rule-based fault prediction: A replication study

Vardis-Dimitris Anezakis  
Democritus University of Thrace,  
Department of Forestry and Management of the Environment and Natural Resources  
Email: danezaki@fmenr.duth.gr

Muhammed Maruf Öztürk  
Suleyman Demirel University, Department of Computer Engineering, Engineering Faculty, Isparta, TURKEY.  
Email: muhammedozturk@sdu.edu.tr

Abstract—The prediction success of faulty modules in a software helps practitioners to plan the budget of software maintenance that leads developers to improve the reliability of software systems. Despite various learning algorithms and statistical methods, fault prediction needs novel methods for enhancing the success of the prediction. Fault prediction can be performed using fuzzy rules that are new for this field. In this work, fuzzy rule-based fault prediction approach, which was developed by Singh et al. [11], is replicated to validate the success of fuzzy rule-based fault prediction in open-source data sets. The steps of the experiment and the steps of Singh et al’s work, which are applied for replication, both are same. Classification is performed after generating clusters that are constituted using fuzzy rules in normalized data sets. According to the prediction results obtained by applying 10*10 cross-validation, fuzzy rule-based fault prediction produces less errors in open-source data sets when it is compared with industrial data sets. In addition to this, the results validate the findings of Singh et al.’s work in terms of some performance parameters of the fault prediction.

Index Terms—Fuzzy rule, fault prediction, software metrics, fault data sets, modulator learning.

I. INTRODUCTION

Nowadays, it is expected that if versions of software are ordered by date, the newest should be the best in terms of reliability and sustainability during software development. Meanwhile, completeness of the processes of a software development reduces the probability of defect occurrence and this helps planning maintenance budget [1]. The increase of the scale of a software and shortened development times require precise software development methods. The more we develop precise prediction methods, the more the defects of a software are detected in early phases of the development. Thus, fault prediction is not only a requirement for developers, but also a facility in terms of the maintenance phase of a software system.

Various metric standards have been developed so far that these standards renew themselves and they change depending on the properties of programming languages. The most widely known and old two metric standards are Halstead [3] and McCabe [2]. With the emergence of object-oriented programming languages, Lorenz and Kidd [4] and Chidamber and Keemer [5] standards were developed. Software metrics include summary values giving information about a software system. These values have hints to developers to decide whether a software module is defective or not as well. Software metrics are extracted from software modules that each of them has a label indicating defectiveness. The value of a label is marked as either 0/1 or true/false. Thus, the defectiveness of a software module is checked by this way. However, the main factor affecting the correctness of a label is the selection of software metrics. Selecting useful metrics helps prevent wrong labelling so that the selection of metrics has become popular among researchers [6], [8], [18], [23], [9], [10].

Fault prediction is not a new working field of software engineering. It dates back to 1990 [33]. A great number of works, which are greater than 100, have been done so far in terms of fault prediction. They comprise class imbalance [16], cross-project fault prediction [34], dealing with noise [35], ensemble learning [18], cost-sensitive learning [36], and so on. However, the works conducted with a fuzzy predictor include defective component prediction, ranking reliability models, and feature selection. It is still unclear to what extent fuzzy predictors are successful with open-source data sets. Singh et al. [11] clustered fault prediction data sets using k-means clustering algorithm. Fuzzy rules were established by means of generated clusters. In their work, it was stated that modulator learning reduced classification error remarkably. The fuzzy rule-based approach outperformed other predictors such as C4.5, random forest, and Naive Bayes.

The main aim of this paper is to replicate Singh et al.’s work [11] to validate their fuzzy rule-based fault prediction approach in real-world data sets. We do apply the steps of their work on open-source fault prediction data sets. Here, the results recorded by them and ours are compared. Comparison is a must to validate the success of fuzzy rule-based approach. Main steps of replication study are seen in Figure 1. The experiment of the paper is conducted to answer following research questions:

Research Question 1 (RQ1): Does fuzzy rule-based fault prediction along with real-world data sets produce promising results as in Nasa MDP data sets? Research Question 2 (RQ2): How the error rates of modulator learning algorithm change depending on the type of the data sets? Research Question 3 (RQ3): Is fuzzy-rule based defect prediction is
competitive against compared predictors? **Research Question 4 (RQ4):** Are the type of metrics of experimental data sets actually prominent in predicting faults with fuzzy-rules?

The rest of the paper is organized as follows. Following section summarizes related works and stresses the need for fuzzy rule-based fault prediction works. Section 3 shows the details of how modulator learning is applied in this study. This section also explains fuzzy rules in depth. Moreover, experimental condition and settings are in Section 3. Section 4 presents experimental results. Last, concluding remarks are given in Section 5.

II. Literature Review

While conducting fuzzy rule-based fault prediction, it is important to decide which type of metrics will be used. Therefore, in this section, predictors and prediction works focusing feature selection are mentioned. Further, we intend to depict the difference of our work and related works by summarizing fuzzy classifiers. By this way, the importance of the paper can be figured out well.

Various predictors including Bayes, Naive Bayes, random forest, and J48 had been used and discussed in preceding works [12], [13], [14]. Each of these predictors has a distinctive property that the decision of which predictor should be used is made by looking the type of the predictor. Therefore, the best classifier for fault prediction is not unique and changes depending on the type of fault prediction data set. For instance, tree-based predictors work very slower than Naive Bayes algorithm [15]. On the other hand, Naive Bayes works on fault prediction data sets along with some statistical assumptions. For instance, Naive Bayes assumes that all features of data sets are statistically independent. In order to increase the success of the prediction, weak classifiers are combined with Boosting methods which have been commonly used in fault prediction for this purpose [16], [18].

Gayatri et al. [17] made feature selection on KC1 data set, which is one of the Nasa MDP data sets, using information gain. The success of the method was evaluated on 18 different classifiers afterwards. Their method can be considered as competitive in terms of AUC (area under the curve) and error rates. However, proposed method was not compared with various feature selection methods.

Feature ordering methods were compared by Gao et al. [19] before. For this comparison, they used search-based algorithms. According to their results, automatic hybrid search yielded best performance among all algorithms. One of the most prominent aspects of this work is that reducing feature space does not create an adverse effect on classification performance. However, the work does not present any detail about selected data set. Further, it has not any comprehensive case study on open-source data sets. These issues are the deficiencies of this work.

Various optimization algorithms are utilized to feature selection. In one of such works [20], instead of than traditional learning algorithms, particle swarm optimization is employed to make feature selection. Instead, bagging technique is in this work to perform classification. Although this work produced promising results, it was not compared with any rule-based classifier.

Shivaji et al. [21] developed an algorithm which sheds new light for researchers to handle with feature selection issue. This algorithm measures the success of both classification and feature selection along with many performance parameters. Despite there are sufficient experimental data sets, the use of two classifiers in this work is a threat for the validity. Besides feature ordering methods, some analyzes such as PCA (principle component analysis) [23] can be used for feature selection.

Clustering is another way to perform feature selection on fault prediction data sets. For instance, the most correlated features were selected using k-means clustering in Kim et al.’s work [22]. In this work, hierarchic clustering outperformed the others in cross-validation experiment. However, the experiment was conducted by using just industrial data sets.

Fuzzy theory is common in developing decision making systems. In one of such works [29], a decision matrix is proposed using fuzzy sets. The findings of this work showed that TOPSIS is competitive with similar group decision-making methods. A similar multiple decision-making problem was presented in [30]. The main purpose of this work is to construct an optimization-based decision maker with the help of fuzzy for project scheduling problem. The method was applied on a hydropower station project and it was strongly depicted in this work that a fuzzy scheduler is suitable for real-life optimization problems. A novel emotion recognition approach was proposed by Halder et al. [31]. They presented IA-IT2FS for constructing membership functions and compared with similar methods. It was stated in this work that fuzzy sets are very sensitive with optimization techniques in classification processes. Fuzzy classifiers had been employed in transportation systems before [32]. In order to control train traction, fuzzy Bayesian model is better than traditional methods in terms of safety, cost and risk management. Fault prediction is a classification problem so that mentioned methods could give
new possibilities to create a distinctive predictor.

A recent study, published by Singh et al., is the most comprehensive fuzzy rule-based fault prediction work to the best of our knowledge [11]. Powerful aspects of this work are the introduction of modulator learning algorithm that increased the success of prediction in Nasa MDP fault prediction data sets and the use of k-means clustering in generating fuzzy rules. Further, their study includes a comparison with other well-known predictors such as Naive Bayes, C4.5 and the scale of the data set is large. Here, we replicate their steps on real-world and open-source data sets. Thus, we do not only compare the performance of fuzzy rule-based systems with other predictors, but also explore the success of their method in real-world and open-source data sets.

III. METHOD
A. The original study selected as replication

Main aim of our study is to replicate the steps of Singh et al’s work [11] by performing a fuzzy rule-based fault prediction. In brief, their work includes five Nasa MDP data sets which are available in tera-promise [37]. It is a widely known repository which is used by software engineering researchers to take or donate research data sets. A z-normalization was applied on experimental data sets. Subsequently, in order to establish fuzzy rules, k-means is selected for clustering. Clustering has been done only training data parts. Modulator learning algorithm was proposed in the original study to increase the accuracy of the prediction. Proposed approach yielded good results which seem to higher than C4.5, random forest, and Naive Bayes in some performance parameters.

One of the originalities of Singh et al’s work is that they expressed defectiveness of a software module using membership functions. Initially, skewed centers of k-means were created to help generating fuzzy rules. Then this obstacle was wiped out with the help of nearest neighbor that instances were assigned to related clusters.

In most current fuzzy rule-based classifiers, the error of the accuracy of the prediction is not reduced well. Proposed modulator learning algorithm found a way to lessen the adverse impact of fuzzy rules. Regarding the firing strength of related rules makes the error of the classification minimum.

Selected comparison predictors are well-known in fault prediction literature. Therefore, the study to be replicated has a sufficient reliability level to perform the replication steps comprehensively. Some points remained unclear that one of them is how successful fuzzy rule-based approach along with open-source data sets might be. This manner can be examined with some statistical measures to reveal the prominence of similar fuzzy rule-based works.

B. Fuzzy notions for fault prediction

In this section, z-normalization, clustering, generation of fuzzy rule base, and experimental conditions are explained. All these processes are also in the work selected as replication. However, the real-world experimental data sets are added to the experiment, rather than using merely Nasa MDP so that some metrics have been changed in fuzzy rules. In order to figure out a framework which was developed using fuzzy rules, notions of software modules in fuzzy terms should be comprehended. This point is detailed in this section.

The space of software modules can be defined as $S = [(x_1, y_1), (x_{i+1}, y_{i+1}), (x_{i+2}, y_{i+2}), ... , (x_n, y_n)]$. In this definition, $S$ is space and $n$ is the number of modules. Given a software module $x_i, y_i$ is the label of defectiveness of the associated software module. Let $A$ be the defectiveness cluster of a software module and membership function of fault prediction conducted with the help of fuzzy rules is in Equation 1. There is a value either 1 or 0 that tells whether the membership is present or not. As the value of membership $\mu_A(x)$ closes to 1, the level of membership $x$ increases. The height of membership functions is 1.

$$\mu_A(x) = \{1 \iff x \in A, 0 \iff x \notin A\} \tag{1}$$

Two fuzzy rule sets are equal in the manner of Equation 2.

$$\forall x \in X, \mu_A(x) = \mu_B(x), \bar{B} = \bar{A} \tag{2}$$

In order to compute the height of a fuzzy logic membership, $tA = \bar{A}^+ := \{x \in X, \mu_A(x) > 0\}$ is utilized. The height of a membership is $ht(A) = t_x \in X \mu_A(X)$.

Let $X = x_1, x_2, x_3, ..., x_n$ denotes software modules, for $\bar{A}$, the membership $\mu_A(x)$ can be expressed as table format. This table gives all the membership degrees of components in space. Equation 3 shows the relationship between a software module and membership degree.

$$\bar{A} = \mu_A(x_1)/x_1 + ... + \mu_A(x_n)/x_n = \sum_{i=1}^{n} \mu_A(x_i)/x_i \tag{3}$$

If $X$ space is infinite, graphic and cube presentations are not sufficient. Instead, the functions in Table I are used for Figure 2.

Generally, software modules are binary classified as defective or not defective so that the type of our membership function is trapezoidal in the experiment. Where $a_1 \leq x \leq a_2$ is near, $a_3 \leq x \leq a_4$ is medium, and $a_2 \leq x \leq a_3$ is far. If a membership degree of a software module is medium then it is labeled as a defective module, other membership degrees are for faultless modules.
TABLE I: Parameter boundaries of trapezoidal membership function.

<table>
<thead>
<tr>
<th>( \mu_A(x) )</th>
<th>( x \in [-1,0] ), ( r, a_1 \leq x \leq a_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{a_2 - r}{a_1 - r} )</td>
<td>( r, a_3 \leq x \leq a_4 )</td>
</tr>
<tr>
<td>( \frac{a_4 - a_3}{a_2 - a_1} )</td>
<td>( r, a_2 \leq x \leq a_3 )</td>
</tr>
<tr>
<td>0, otherwise</td>
<td></td>
</tr>
</tbody>
</table>

C. Construction of Rule Base

Initially, the data sets, which are selected for replication, are exposed to z-normalization. Z-normalization is performed to make each feature has same standard deviation. This type of normalization is to restrict the boundary of values between mean zero and standard deviation. Mean zero is ignoring total irregularities. The scaling factor is standard deviation. Thus, extreme values feature a great effect on composite indicator. Z-score ordering is calculated with (real value-mean criteria)/criteria standard deviation. We have used R package to perform normalization. One of the chief challenges of normalizing data is that normalization could not be completed at one time with multi-feature. Instead, normalization was conducted by each feature independently. This increases the completion time of normalization remarkably that such operation is effort-intensive for fault prediction data sets.

After assigning new values between -1 and 1, if some values are still greater than 1 after z-normalization, they are equalized to threshold value (0.8). This threshold value is the mean of the values which are close to 1 in normalization. In order to perform clustering, k-means clustering is selected which was also used in original study. We have clustered experimental data sets in 10 clusters with this algorithm. A table is then generated that it includes the number of sample, cluster number, and defect labeling true/false. Fuzzy rules are established by looking the ratios of true/false of 10 clusters. The highest true/false rate is used for deciding defective sample bias to establish associated fuzzy rules. Likewise, the lowest true/false rates are used for deciding not defective sample bias to establish associated fuzzy rules.

Clustered instances can be expressed as follows. Given an instance \( S = x_1, ..., x_n \) and cluster set \( C = c_{1,j_1}, ..., c_{n,j_n} \), if \( x_i \) is close to \( c_1 \), the label of \( x_i \) is \( c_{j_1} \). If we define these notions as fuzzy rules, let \( M = m_1, m_2, m_3, ..., m_n \) denotes metric values and threshold values to be used for them are \( a, b, ..., k \). The rule emerges as \( R_1 \) : if \( x_{m1} \) is close to \( a, ..., n \), \( x_{mn} \) is close to \( k \) then the class is true/false. Though feature count is sufficient to create fuzzy rules in initial condition, it decreases with feature selection. Following section summarizes modulator learning algorithm which is fundamental in the work selected as the basis of our replication study.

D. Modulator Learning Algorithm

The main function of modulator learning algorithm, which is the contribution of Singh et al’s work, is of twofold. First is the computation of error rates of fuzzy rules. Second is the reduction of classification error. The reason why this algorithm is the contribution of Singh et al.’s work is that with the help of this algorithm, fuzzy rule-based classifier outperformed other three predictors. It can be detailed in four steps.

First step: Initially the boundaries of fuzzy membership function are specified. Trapezoidal membership function is selected. It is determined that which ranges produce defectiveness afterwards.

Step two: After adding fuzzy rules to inference system, firing strengths of all the rules and membership values \( \mu(x) \) are calculated. Second step includes the creation of fuzzy rule data base and the computation of \( E^x \) function for all instances.

Step three: The skewness of classification \( M^x \) is computed. The computation of \( E^x \) is performed until the minimum classification error is reached.

Step four: The computation is suspended if the value of \( E^x \) or \( M^x \) is equal to zero.

\( E^x \) error function is defined as in Equation 4.

\[
E^x = \sum_{x \in X} E_x = \sum_{x \in X} \sum_{l=1}^{c} (o_l - t_l)[11] \tag{4}
\]

For \( n \) instances, the firing strength of the relevant rule is expressed with \( a_{ji} = T_{k=1}^{p}(\mu_{k,jx}) \). The max of \( a_{ji} \) values gives \( o_l \).

E. Data Sets

Two groups of data sets are used in experimental study. The first of them is Nasa MDP data sets which were used in Singh et al.’s work [11]. These data sets are commonly used industrial fault prediction data sets retrieved from tera-promise repository [25]. Second group is real-world data sets extracted from open-source software projects [24]. In the experiment, 5 out of 11 data sets are open-source. Thus, the success of fuzzy rule-based approach in industrial data sets has also been explored in open-source data sets.

Note that experimental data sets were extracted from some projects coded by three different programming languages. Therefore, the static code attributes of these data sets differ. Inherently, the fuzzy rules from Singh et al. [11] and the fuzzy rules created in our study are completely different. All these are elements supporting the validity of our study. All the metrics of LOC and Halstead are in cm1, jm1, kc1, kc2, and kc3. Some part of McCabe is also available in these data sets.

F. Experimental Conditions

1) Experimental Environments: All the experiments have been completed on a machine which has Windows 7 operating system, 64-bit, Intel Xenon 3.1Ghz processor, and 16 GB ram. Fuzzy rule-based system has been coded with C# in Visual Studio 2013 development IDE. AForge.NET library is utilized in some parts of the codes [27]. Besides, in order to draw membership functions and compare \( E(x) \) values, “sets” library from R package is used. Moreover, z-normalization has also been completed with R package. K-means clustering is done with Accord.MachineLearning library [26].
2) Performance Parameters: We have selected some performance parameters which had been employed in original study. One of them is the comparison of $E(x)$ error values of open-source and industrial data sets. Fuzzy rule-based classification performance of open-source data sets has been recorded and evaluated in terms of area under the curve (AUC). AUC is the area under the ROC. ROC is created by drawing true positive rate against false positive rate that their formulas are presented in Table III. It is a good indicator that if AUC is close to 1. Our experiments also do record specificity and accuracy on open-source data sets.

3) Clusters and rules: Open-source data sets are divided into 10 different clusters by using k-means clustering. T/F (true/false) rates have helped to establish fuzzy rules from relevant instances. The highest T/F value is for the rules describing the module as defective and the lowest T/F rate is for the rules describing the module as not defective.

K-means clustering results of open-source data sets are seen in Table IV. When the ratios of T/F are examined, it can be seen that all the columns have two bold-faced values. These values specify the clusters to be selected for fuzzy rules. Whilst low value is used to select the cluster for deciding not defective instances, high value shows that it’s cluster will be used for defective instance.

In order to generate $E(x)$ values from modulator learning algorithm and fuzzy rules from clusters, numberOfLinesOfCode, lcom, rfc, and wmc have been selected. The membership functions of these metrics are demonstrated in Figure 3, Figure 4, Figure 6, and Figure 5, respectively. Using these membership functions, five fuzzy rules have been created.

The details of established fuzzy rules are seen in Table V. These are rules derived from all the datasets by which each rule has the average values of them.

According to these rules, Rule 1 asserts that if $\text{LOC(numberOfLinesOfCode)}$ is medium, $\text{rfc}$ is medium, and wmc is medium then the instance is defective. Likewise, Rule 2 works using same way but it has different membership boundaries. Rule 3 asserts that if $\text{LOC}$ is near, lcom is near, rfc is near, wmc is far then the instance is not defective. It is stated in Rule 4 that if $\text{LOC}$ is medium, lcom is near, rfc is near, and wmc is near then the instance is not defective. Last, Rule 5 asserts that if $\text{LOC}$ is near, lcom is near, rfc is near,
TABLE IV: T/F rates of clusters of data sets.

<table>
<thead>
<tr>
<th>Cluster No</th>
<th>T/F for jdt core</th>
<th>T/F for eclipse pde</th>
<th>T/F for equinox</th>
<th>T/F for lucene</th>
<th>T/F for mylyn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.12</td>
<td>0.04</td>
<td>0.56</td>
<td>0.09</td>
<td>0.28</td>
</tr>
<tr>
<td>1</td>
<td>0.09</td>
<td>0.75</td>
<td>1</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>0.08</td>
<td>1</td>
<td>0.62</td>
<td>0/1</td>
<td>0.08</td>
</tr>
<tr>
<td>3</td>
<td>1.46</td>
<td>0.16</td>
<td>0.47</td>
<td>0/6</td>
<td>0.03</td>
</tr>
<tr>
<td>4</td>
<td>0.26</td>
<td>0.15</td>
<td>0.45</td>
<td>0.4</td>
<td>0.05</td>
</tr>
<tr>
<td>5</td>
<td>0.41</td>
<td>0.57</td>
<td>0.58</td>
<td>1/0</td>
<td>0.25</td>
</tr>
<tr>
<td>6</td>
<td>0.075</td>
<td>0.07</td>
<td>1.3</td>
<td>0/3</td>
<td>0.12</td>
</tr>
<tr>
<td>7</td>
<td>2.26</td>
<td>0.18</td>
<td>0.86</td>
<td>0/4</td>
<td>0.05</td>
</tr>
<tr>
<td>8</td>
<td>0.25</td>
<td>0.16</td>
<td>0.84</td>
<td>0.3</td>
<td>0.22</td>
</tr>
</tbody>
</table>

TABLE V: Fuzzy-rules used in this study.

<table>
<thead>
<tr>
<th>Rule Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>If numberOfLinesOfCode is CLOSE TO 200 and lcom is CLOSE TO 180 and rfc is CLOSE TO 70 and wmc is CLOSE TO 40 then the code is Faulty.</td>
</tr>
<tr>
<td>2</td>
<td>If numberOfLinesOfCode is CLOSE TO 210 and lcom is CLOSE TO 170 and rfc is CLOSE TO 98 and wmc is CLOSE TO 49 then the code is Faulty.</td>
</tr>
<tr>
<td>3</td>
<td>If numberOfLinesOfCode is CLOSE TO 98 and lcom is CLOSE TO 6 and rfc is CLOSE TO 36 and wmc is CLOSE TO 72 then the code is not Faulty.</td>
</tr>
<tr>
<td>4</td>
<td>If numberOfLinesOfCode is CLOSE TO 140 and lcom is CLOSE TO 45 and rfc is CLOSE TO 22 and wmc is CLOSE TO 3 then the code is not Faulty.</td>
</tr>
<tr>
<td>5</td>
<td>If numberOfLinesOfCode is CLOSE TO 26 and lcom is CLOSE TO 3 and rfc is CLOSE TO 11 and wmc is CLOSE TO 7 then the code is not Faulty.</td>
</tr>
</tbody>
</table>

IV. RESULTS

This section discusses the research questions presented in Section I. Firstly it should be stated that all the predictions in the experiment have been performed using 10*10 cross-validation. This type of validation is a common and preferable method in fault prediction works.

RQ1: Table VIII shows AUC values of open-source experimental data sets. The values of AUC of these data sets are higher than 0.5 except recorded from equinox. Equinox has 324 instances that is the lowest value among all the data sets. This case may have affected the result. In addition, it can be concluded that fuzzy rule-based fault prediction yields promising results in open-source data sets as in industrial data sets.

RQ2: In fuzzy rule-based classification, open-source and industrial fault prediction data sets are compared in terms of E(x). This is also an analysis giving valuable information which shows how modulator learning algorithm behaves in open-source data sets. E(x) results of open-source and Nasa MDP data sets are illustrated with box-plots as in Figure 7 and Figure 8, respectively. When we look into these results, it is clear that E(x) values of Nasa MDP data sets are above 2. On the other hand, E(x) values are below 2 in open-source data sets. The results showed that modulator learning algorithm is more suitable when open-source data sets are used instead of industrial ones.

RQ3: As seen from the results presented in Table 7, fuzzy rule-based fault prediction approach relatively better than the other predictors in terms of specificity and accuracy. However, this situation may be change if a general evaluation is made via AUC. Here, optimal cut-off is selected as 0.6. This value has been obtained making various iterations on the data sets. The comparison predictors are well-known techniques that are in common use. Therefore, the reliability of the comparison results is very high. In our replication study, fuzzy rule-based approach is better than J48, Naive Bayes, and random forest that Singh et al. had found similar findings. Fuzzy rule-based fault prediction has the potential that it should be further explored.

RQ4: Fuzzy rule-based fault prediction yielded promising results in five different open-source data sets. The metrics used in the experiment are quite different from the original study that inherently, the fuzzy rules of two studies are also different. The AUC results presented in Figures 9-11 are not greater than 0.66. Nasa Promise data sets could produce higher AUC than open-source data sets. The type of the metric sets may have affected this case.

TABLE VII: Delta estimate results of effect size of the one-way ANOVA with 95 percent confidence interval.

<table>
<thead>
<tr>
<th>Fuzzy-rule-J48</th>
<th>Fuzzy-rule-Naive Bayes</th>
<th>Fuzzy-rule-Random Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.52(large)</td>
<td>0.6(large)</td>
<td>0.4(medium)</td>
</tr>
</tbody>
</table>

TABLE VII: AUC results of the data sets.

<table>
<thead>
<tr>
<th>Data set</th>
<th>AUC</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>mylyn</td>
<td>0.62</td>
<td>0.02</td>
</tr>
<tr>
<td>lucene</td>
<td>0.61</td>
<td>0.03</td>
</tr>
<tr>
<td>jdtCore</td>
<td>0.66</td>
<td>0.02</td>
</tr>
<tr>
<td>equinox</td>
<td>0.49</td>
<td>0.02</td>
</tr>
<tr>
<td>eclipsePde</td>
<td>0.61</td>
<td>0.02</td>
</tr>
</tbody>
</table>
V. CONCLUDING REMARKS

In this work, we explore Singh et al.’s fuzzy rule-based fault prediction approach on open-source fault prediction data sets. Their work had produced good results in terms of both generating fuzzy rules and selecting useful features. They recorded best results in fuzz rule-based classifier when it is compared with the other predictors. Z-normalization and k-means processes have been replicated using modulator learning algorithm. In the experiment, AUC, sensitivity, and specificity are selected for the evaluation of fuzzy-rule based fault prediction.

The results of modulator learning algorithm are better in open-source data sets when it is compared with other results recorded in industrial data sets in terms of $\mathbf{E}(x)$ (Figure 6, Figure 7). Hence, it can be stated that fuzzy rule-based approach is much suitable for open-source data sets. However, it is not negligible that the feature set of fuzzy rules affect this situation. Two main issues make threat for the generality of fuzzy rule-based fault prediction approach. First, using new clustering methods, which were released in this decade, such as k-means++ may give highly accurate results. Second, the number of the clusters is restricted with 10 that currently it is ambiguous how many clusters should be used in such experiments. These two issues can be tested on varied experimental environment. Predictors have been compared with one-way ANOVA to reveal the difference. In our analysis it has been found that all the predictors are different ($p < 0.05$). However, it should be detected that what is the degree of this difference. To compute this, we used effect size statistical measure. This method tells the degree of a statistical difference in three levels: small, moderate, and large. The interpretation of this method namely Cohen’s d was first presented by Cohen in 1988 [28]. A degree is given to the difference by looking the value of Cliff’s delta. Table 8 shows that the difference between classifiers is quite large. In this table, it is evident that a decline is in fuzzy rule-random forest comparison that its degree of the difference is medium. In summary, fuzzy rule-based predictor is superior for detecting faulty instances.
The main reason of this manner is that classification error is minimized along with modulator learning in fuzzy rule-based approach. As a result, using the most effective features, the best rules have been established. It is hoped that the results of this work could make a profound change on traditional fault prediction methods.

REFERENCES