IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 7, 2019, accepted April 29, 2019, date of publication May 9, 2019, date of current version May 24, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2915964*

# Applying the FAHP to Improve the Performance Evaluation Reliability of Software Defect Classifiers

**HUSSAM GHUNAIM AND JULIUS DICHTER**
School of Engineering, Computer Science and Engineering Department, University of Bridgeport, Bridgeport, CT 06604, USA
Corresponding author: Hussam Ghunaim (hghunaim@my.bridgeport.edu)

**ABSTRACT** Today's software complexity makes developing defect-free software almost impossible. Consequently, developing classifiers to classify software modules into defective and non-defective before software releases have attracted great interest in academia and software industry alike. Although many classifiers have been proposed, no one has been proven superior over others. The major reason is that while a research shows that classifier A is better than classifier B, we can find other research that shows the opposite. These conflicts are usually triggered when researchers report results using their preferable performance evaluation measures such as, recall and precision. Although this approach is valid, it does not examine all possible facets of classifiers performance characteristics. Thus, the performance evaluation might improve or deteriorate if researchers choose other performance measures. As a result, software developers usually struggle to select the most suitable classifier to use in their projects. The goal of this paper is to apply the fuzzy analytical hierarchy process (FAHP) as a popular multicriteria decision-making technique to reliably evaluate classifiers' performance. This evaluation framework incorporates a wider spectrum of performance measures to evaluate classifiers performance rather than relying on selected preferable measures. The results show that this approach will increase software developers' confidence in research outcomes and help them in avoiding false conclusions and infer reasonable boundaries for them. We exploited 22 popular performance measures and 11 software defect classifiers. The analysis was carried out using KNIME data mining platform and 12 software defect data sets provided by the NASA metrics data program (MDP) repository.

**INDEX TERMS** Classifiers, data mining, empirical software engineering (ESE), FAHP, KNIME, performance evaluation, software defect.

## I. INTRODUCTION

Software defects are a serious threat to the success of software development industry [1]. In average, billions of dollars are lost every year because of software defects in the United States alone [2], where the global loss is much larger than this amount. Although defects can be detected by various quality procedures, finding and fixing defects consume a significant portion of the available resources [3]. Most software defects are normally found within a relatively small number of modules [4], [5]. Therefore, developing software defect classifiers has become a promising methodology to identify defective modules before software release. The expected returns are significant in terms of reducing the overall quality assurance activities time and costs [1], [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Vijay Mago.

The major aim of software defect classifiers is to classify software modules into defective *dM* and non-defective *ndM*. This binary classification can be described as a mapping function from a vector $\boldsymbol{x}$ of $M$ features, where $\boldsymbol{x}_i \in R^M$, to one of the classification classes $y_i \in \{dM,\ ndM\}$ [4].

$$f(\boldsymbol{x}) : R^M \mapsto \{dM,\ ndM\} \tag{1}$$

This model can be trained by a training data set $S$ of $N$ instances,

$$S = \{(\boldsymbol{x}_i,\quad y_i)\}_{i=1}^{N}. \tag{2}$$

Numerous techniques have been proposed to develop classifiers, for instance, regression and logistic regression, neural networks, decision trees, and many other machine learning algorithms [4], [7] with no superiority of any over the rest [3], [8]. This is mainly caused by conflicted benchmarking studies. Various software engineering research papers [1], [3], [9], [10] investigated and challenged the reliability of software defect classifiers benchmarking studies. The common finding among these studies was that while one-study showed classifier A was better than classifier B, we could find other studies that showed exactly the opposite.

Consequently, software practitioners face the problem of how they can reliably evaluate the performance of defect classifiers to select the best performing classifier out of several others [11]. Although there are many performance evaluation measures, they usually provide contradicting results. This contradiction is indeed expected as each of these measures was developed to capture a specific aspect of classifiers' performance. For example, recall, which is known as True Positive Rate (TPR), represents the proportion of the actual defective modules that are classified defective. Similarly, precision, which is known as Positive Predictive Value (PPV), represents the proportion of classified defective modules that are actually defective [3], [12], and so forth. As a result, the performance quality is highly dependent on the specific measures being utilized!

This fact leads to the critical question; which performance evaluation measure(s) practitioners should use? In other words, how practitioners can evaluate classifiers in such a way that they will always get reliable and valid results? This essential requirement is motivated by two possible scenarios. Mistakenly classifying defective modules as non-defective raises the risk of software failure. While classifying non-defective modules as defective raises software quality assurance activities time and costs. We believe that the proposed evaluation framework will increase the reliability and researchers' confidence in the classifiers' evaluation process outcomes.

This paper is organized as follows, related work Section discusses the problem of classifiers' evaluation reliability and its implications on software development industry. Section III discusses the principles of Fuzzy Analytical Hierarchy Process (FAHP). The classifiers' performance evaluation measures implemented in this research are presented and explained in Section IV. In data sets Section, we describe several data quality issues that exist in NASA data sets and the cleaning procedures that are performed. In experimental setup design Section, we discuss how the experiments and analysis are performed while Section VII, the FAHP Application, describes in detail how we applied FAHP to the classifiers' performance evaluation. The results Section discusses the paper results and conclusion Section summarizes the conclusions.

## II. RELATED WORK

The reliability of software defect classifiers was scrutinized extensively in many published works [8], [11], [13], [14], [15]. Nonetheless, it seems that there is a large room for improvements. For example, performance quality measures, such as, precision, accuracy, etc. can be improved by applying rigorous reliability and verification techniques [8], [11]. Additionally, many of these measures are borrowed from other disciplines (e.g. Psychology and social sciences). In many cases when these measures are used 'as is', they usually have different implications [12].

It has become a common practice for practitioners and researchers to select their most preferable statistics to support their point of view. Consequently, vague and misleading conclusions might result. Forman and Scholz [16] concluded in their study that comparing different research studies has become complicated, and in many cases, the comparisons have become not meaningful.

This paper emphasizes the fact that performance evaluation must be seen as a comprehensive strategy rather than relying on preferably selected performance measure(s). Shepperd *et al.* [3] conducted an extensive study to find the reasons of variance in classifiers performance. Their study included 600 experimental results published in many reputable conferences and journals with low acceptance rates. Surprisingly, researcher bias was among the major and widespread influential factors. They found that it is extremely difficult to choose the best performing classification technique because of this phenomenon.

To solve the problem of researchers' bias, Inse *et al.* [17] asserted that researchers should improve their research outcomes reporting protocols. Kitchenham [18] also suggested the need to enhance the communication and documentation protocols to include sufficient explicit details about how exactly classifiers were used and evaluated in research.

Fenton and Bieman [19] extensively discussed the concept of research reliability. In general, he emphasized the empirical validity procedures where researchers are required to validate their findings by replications of experiments. Empirical validation studies have become an essential part in software defect classification research because usually we lack the required theoretical validation. This fact has led us to our paper contribution which is proposing a comprehensive evaluation scheme that will provide proven better evaluation outcomes compared to preferred selected performance measure(s).

## III. FUZZY ANALYTICAL HIERARCHY PROCESS (FAHP)

To avoid the researchers' bias when evaluating the performance of software defect classifiers, this paper proposes the application of multicriteria decision making (MCDM). MCDM is a set of very effective methodological tools for dealing with complex problems in various domains, such as, business, engineering, etc. Some examples are, AHP, FAHP, TOPSIS, and many others [20]–[22].

The Analytical Hierarchy Process AHP technique is based on expert judgment method to perform pairwise comparisons between all implemented criteria. However, AHP suffers from a critical criticism. AHP is unable to deal with the impression and subjectiveness of the expert judgment when performing the pairwise comparisons method [23]–[25].

In recent years, Fuzzy AHP–or for short, FAHP–gained a noticeable attention as a superior substitute to the AHP technique. The essence of FAHP method is based on the

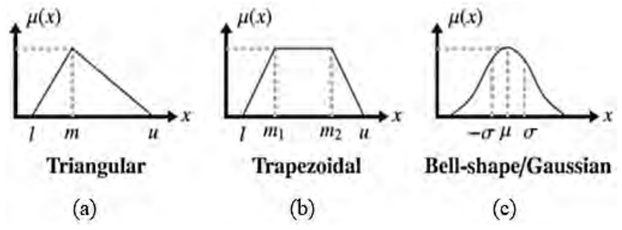

**FIGURE 1.** Membership functions used in FAHP.

ability to capture the uncertainty when performing expert judgment method. Zadeh [26] introduced the fuzzy set theory to compromise the human thought vagueness, which was oriented to the rationality of uncertainty due to imprecision or vagueness. That is, the consideration of the gradual membership of an element to a particular set of elements [23].

Peng [27] and Kabir and Hasin [28] applied AHP successfully in the field of classifiers performance evaluation. In this paper, the authors apply FAHP in evaluating binary classifiers performance as a more robust multicriteria procedure. Up to our knowledge, this is the first time of such application.

In 1983, Laarhoven, et al. proposed the use of a triangular fuzzy membership function as the best fit in performing expert judgment, Fig (1.a) [29]. Other functions were proposed as well to fit various uses, Fig (1.b and 1.c). We chose to use the triangular membership method for its suitability to software defect classifiers domain, Equation (3). The reason for this choice is that we need only to provide two boundaries to our judgment, upper and lower boundaries when comparing measures pairwise. Trapezoidal function for example provides two middle values in addition to the upper and lower boundaries. The is not the case in our research. Similar arguments are applicable to other fuzzy membership functions that might require unnecessary complications. Thus, for simplicity, we made this choice.

$$\mu(x|\widetilde{M}) = \begin{cases} 0, & x < l \\ (x-l)/(m-l), & l \le x \le m \\ (u-x)/(u-m), & m \le x \le u \\ 0, & x > u \end{cases} \tag{3}$$

Throughout this paper, fuzzy quantities are differentiated by placing a tilde '$\sim$' above symbols. A triangular fuzzy number (TFN) is denoted as $(l, m, u)$, where $l$ denotes the smallest possible value, $m$ denotes the most promising value, and $u$ denotes the largest possible value that describes a fuzzy event. Readers interested in more detailed introduction to fuzzy numbers and their algebraic operations are recommended to read Harding *et al.* [30].

## IV. PERFORMANCE EVALUATION MEASURES

To evaluate the classifiers' performance, we followed the common practice of using a confusion matrix, Table 1. The first column shows the actual positive **(AP)** cases (defective modules) and the second column shows the actual negative **(AN)** cases (non-defective modules). Similarly, the first row shows the predicted positives (PP) and the second row shows the predicted negatives (PN). The bottom right cell shows the total number of cases, ***T***. While the optimum desired results would be $fp = fn = 0$, the actual performance of classifiers is still far away from achieving this goal. By utilizing these four variables, the classifiers' performance measures can be calculated.

**TABLE 1.** Confusion matrix.

| | | |
|---|---|---|
| tp | fp | **PP** |
| fn | tn | **PN** |
| **AP** | **AN** | ***T*** |

Numerous performance measures have been proposed and utilized by researchers and practitioners to evaluate classifiers' performance. Table 2 shows the 22 performance measures exploited in our research [3], [27]–[33]. The selection of these measures was based on their popularity in software defect classification research [3], [12]. Since Cohen's Kappa is the only measure that needs more clarifications on how to compute its probabilities (i.e. $\Pr(a)$ *and* Pr(e)), we added those clarifications right after the table.

## V. DATA SETS

As the requirement of research replication has become vital for many researchers, we have decided to use the publicly available and widely used NASA software defect data sets [34]. One justifiable reason that explains this choice is to support the ability to reproduce and verify the published results, and to ease data sharing among researchers [35].

However, NASA data sets suffer from many data quality problems [36]. For clarity, we repeat here the common assumptions about software data sets structure. NASA data sets are organized as a matrix of rows and columns. Each row represents one software module (i.e. case), and each column represents one feature (i.e. attribute). One of the data integrity issues that can be found in NASA data sets is the identical values. This problem occurs when two or more features have the same values for all cases. Similarly, if two or more cases have the same values for all features including the prediction label. Identical features present no additional information and identical cases confuse learners, thus, both were excluded from data sets. Conflicting values is another problem that arises whenever there is a violation of a relational integrity constraint. For example, LOC_TOTAL cannot be less than LOC_EXECUTABLE or LOC_COMMENTS. Fan et al. [37] have discussed integrity constraints in more details. Such data is untrustworthy and should not be included in the data sets. Additionally, implausible values are any negative or fractional values that do not make sense and cannot be accepted. Similar to conflicting values, this data is untrustworthy. Cases' inconsistency, constant values and missing values were among other data integrity issues that were analyzed.

**TABLE 2. List of the 22 performance evaluation measures exploited in the study.**

| | | |
|---|---|---|
| 1 | Recall | $= tp/(tp+fn)$ |
| 2 | Precision | $= tp/(tp+fp)$ |
| 3 | Inverse Recall | $= tn/(tn+fp)$ |
| 4 | Inverse Precision | $= tn/(tn+fn)$ |
| 5 | Area Under ROC Curve AUC | = (Recall + Inverse Recall)/2 |
| 6 | Accuracy ACC | = (tp + tn)/(tp + fp + tn + fn) |
| 7 | F1-Score | $= 2tp/(2tp+fn+fp)$ |
| 8 | Informedness | $= Recall + Inverse\ Recall - 1$ |
| 9 | Markedness | $= Precision + Inverse\ Precision - 1$ |
| 10 | Matthews Correlation Coefficient MCC | $= \frac{tp \times tn - fp \times fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}}$ |
| 11 | G-Mean1 | $= \sqrt{Recall \times Precision}$ |
| 12 | G-Mean2 | $= \sqrt{Recall \times Inverse\ Recall}$ |
| 13 | Cohen's Kappa ***(further details below)*** | = (Pr(a) − Pr(e))/(1 − Pr(e) ) |
| 14 | False Discovery Rate (FDR) | $= fp/(fp+tp)$ |
| 15 | False Omission Rate (FOR) | $= fn/(fn+tn)$ |
| 16 | False Positive Rate (FPR) | $= fp/(fp+tn)$ |
| 17 | False Negative Rate (FNR) | $= fn/(fn+tp)$ |
| 18 | Predicted Positive Condition Rate | $= (tp+fp)/(tp+tn+fp+fn)$ |
| 19 | Positive Likelihood Ratio (LR+) | $= Recall/FPR$ |
| 20 | Negative Likelihood Ratio (LR−) | $= FNR/(Inverse\ Recall)$ |
| 21 | Diagnostic Odds Ratio (DOR) | $= (LR+)/(LR-)$ |
| 22 | Prevalence | $= (tp+fn)/(tp+tn+fp+fn)$ |

Cohen's kappa probabilities are calculated as follows:

- $\Pr(a)$ : is the observed agreement probability among raters $= \frac{tp+tn}{tp+fp+tn+fn}$
- $\Pr(e)$ : is the agreement by chance probability among raters $= R_1(P)\,R_2(P) + R_1(N)\,R_2(N)$
- Rater1 percentage of positive responses

$$R_1(P) = \frac{tp+fp}{tp+fp+tn+fn}$$

- Rater1 percentage of negative responses

$$R_1(N) = 1 - R_1(P)$$

**TABLE 3. Changes made to NASA data sets after applying the cleaning strategy. ∗df% is the percentage of defective modules.**

| NASA Data Sets | Original Data sets | | | Cleaned-up Data sets | | |
|---|---|---|---|---|---|---|
| | #Modules | #Attributes | df % * | #Modules | #Attributes | df % * |
| CM1 | 505 | 41 | 9.50 | 327 | 38 | 12.84 |
| JM1 | 10878 | 22 | 19.32 | 7782 | 22 | 21.49 |
| KC1 | 2107 | 22 | 15.42 | 1183 | 22 | 26.54 |
| KC3 | 458 | 41 | 9.39 | 194 | 22 | 18.56 |
| MC1 | 9466 | 40 | 0.72 | 1988 | 22 | 2.31 |
| MC2 | 161 | 41 | 32.30 | 125 | 22 | 35.20 |
| MW1 | 403 | 41 | 7.69 | 253 | 22 | 10.67 |
| PC1 | 1107 | 41 | 6.87 | 705 | 22 | 8.65 |
| PC2 | 5589 | 41 | 0.41 | 745 | 22 | 2.15 |
| PC3 | 1563 | 41 | 10.24 | 1077 | 22 | 12.44 |
| PC4 | 1458 | 41 | 12.21 | 1287 | 22 | 13.75 |
| PC5 | 17186 | 40 | 3.00 | 1711 | 22 | 27.53 |

- Rater2 percentage of positive responses

$$R_2(P) = \frac{tp+fn}{tp+fp+tn+fn}$$

- Rater2 percentage of negative responses

$$R_2(N) = 1 - R_2(P)$$

Shepperd *et al.* [34] and [36] performed a comprehensive cleaning strategy to remove all problematic cases and features, Table 3. They published the cleaned-up data sets resulted after removing all cases and features that had one or more of the discussed data quality problems. These data sets are available online at
https://figshare.com/collections/NASA_MDP_Software_Defects_Data_Sets/4054940/1.

## VI. EXPERIMENTAL SETUP DESIGN

We exploited eleven software defect classifiers, Table 4. These classifiers have been chosen based on their popularity in software defect research [4], [38]. The experiments were carried out using KNIME [39], [40], a popular data mining platform and twelve NASA software defect data sets.

KNIME was used to run the classifiers on all experimented data sets. The corresponding confusion matrixes were constructed and utilized to calculate the classifiers' performance measures, that is, $E[c \times p]$ matrixes, where $c$ is the number of classifiers and $p$ is the number of performance measures. To validate the results, 10-fold cross-validation technique was run on all experiments. Additionally, we normalized all experimented data sets to avoid the dominance of some attributes with large values.

Imbalanced data sets can degrade classifiers performance and contribute to the results unreliability [35], [54]. It is quite common in software defect data sets to have non-defective

**TABLE 4.** Software defect classifiers.

| | |
|---|---|
| 1 | Probabilistic Neural Network (PNN) based on the Dynamic Decay Adjustment (DDA) [41] |
| 2 | (SOTA) clustering [42] |
| 3 | Fuzzy Rule (FR) [43] |
| 4 | Logistic Regression (LR) [44] |
| 5 | Naïve Bayes (NB) [45] |
| 6 | K Nearest Neighbor (KNN) [46] |
| 7 | Multi-Layer Perceptron (MLP-RProp) [47] |
| 8 | Support Vector Machine (SVM) [48] [49] |
| 9 | Decision Tree C4.5 (DT) [50] [51] |
| 10 | SimpleCart (CART) [52] |
| 11 | Random Forest (RF) [53] |

modules as the majority class while the defective modules as the minority class. Therefore, stratified sampling technique was used to avoid sampling bias. Stratified functionality guaranteed that all created cross-validation folds had class distribution similar to the original data sets distributions, i.e. the ratio of defective to non-defective modules.

For clarity, we start with presenting a summary of the FAHP steps implemented in this study followed by more detailed calculations in the following section, FAHP Application.

**Note**: *Matrixes are denoted by italicized capital letters, and vectors are denoted by bold face italicized small letters.*

```
Let,
$c = 11$, $c$ is number of classifiers,
$p = 22$, $p$ is number of performance measures,
$d$ data set
$D$ the set of 12 NASA data sets
1) Construct the fuzzy performance measures pairwise
   comparisons $\tilde{A}[p \times p]$ matrix.
2) Compute the criteria fuzzy weight vector $\tilde{\boldsymbol{w}}$ from
   $\tilde{A}$ matrix.
3) Compute the defuzzified criteria weight vector $\boldsymbol{w}$.
 for each $d \in D$ do
    4) Compute the classifiers evaluation matrix $E[c \times p]$.
    5) Compute the classifiers scores $S[c \times p]$ matrix.
        a. compute $p$ number of $B^{(j)}$ matrixes
           (classifiers pairwise comparisons) with respect
           to each criterion $j = 1, \ldots, p$
        b. from each $B^{(j)}$, compute $\boldsymbol{s}^{(j)}$ score vector
        c. construct the $S[c \times p]$ matrix by combining
           all $\boldsymbol{s}^{(j)}$ vectors, column wise.
    6) Compute the classifiers ranking $\boldsymbol{v} = S \cdot \boldsymbol{w}$,
       where $\boldsymbol{v}_i$of the vector $\boldsymbol{v}$ represents the global score
       (i.e. rank) assigned by the FAHP to the $i^{th}$ classifier.
    7) Identify the highest performing classifier compared
       to the list of experimented classifiers.
end for
```

**TABLE 5.** AHP and FAHP scores interpretations.

| AHP Crisp Scale $a_{jk}$ | FAHP TFN $(l, m, u)$ $\tilde{a}_{jk}$ | Interpretation $j$ and $k$ denote criteria |
|---|---|---|
| 9 | 9,9,9 | $j$ is extremely more important than $k$ |
| 7 | 6,7,8 | $j$ is strongly more important than $k$ |
| 5 | 4,5,6 | $j$ is more important than $k$ |
| 3 | 2,3,4 | $j$ is slightly more important than $k$ |
| 1 | 1,1,1 | $j$ and $k$ are equally important |
| 1/3 | 1/4,1/3,1/2 | $j$ is slightly less important than $k$ |
| 1/5 | 1/6,1/5,1/4 | $j$ is less important than $k$ |
| 1/7 | 1/8,1/7,1/6 | $j$ is strongly less important than $k$ |
| 1/9 | 1/9,1/9,1/9 | $j$ is extremely less important than $k$ |

## VII. FAHP APPLICATION

The following are the FAHP implementation steps [23], [27]:

**Step 1:**

Decompose the problem into three hierarchical levels, Fig. (2).

- **Goal**: evaluating the performance of software defect classifiers to select the best performing classifier
- **Criteria**: twenty-two performance measures
- **Alternatives**: eleven software defect classifiers

**Step 2:**

Perform fuzzy pairwise comparisons between all criteria elements using the fundamental scale proposed by Saaty [22], Table 5. At the end of this step, a *criteria fuzzy weights vector* $\tilde{\boldsymbol{w}}$ is computed. However, this scale is based on crisp evaluation values. As discussed in Section III, crisp evaluation usually leads to unreliable results due to the expert judgment uncertainty and vagueness. Thus, the scale must be modified to meet FAHP requirements. That is, instead of evaluating the criteria using the crisp scale values, we can use the Triangular Fuzzy Numbers (TFN) to compromise for human uncertainty and increase the reliability of the evaluation. Note that for any fuzzy number $\tilde{a}$, the reciprocal can be defined as

$$\tilde{a}^{-1} = (l, m, u)^{-1} = \left(\frac{1}{u}, \frac{1}{m}, \frac{1}{l}\right) \tag{4}$$

Table 5 entries are only suggestive to translate the decision maker qualitative evaluations of the criteria into quantitative values. It is possible to use other similar scales.

The authors use their extensive experience in the field of binary classifiers evaluation measures to rank their relative importance following Saaty fundamental scale of weights. Additionally, the literature provides a large body of research to evaluate the reliability and validity of each of these measures. For brevity, a representing sample is cited in this paper [13], [33], [12]. Table 6 shows the relative fuzzy

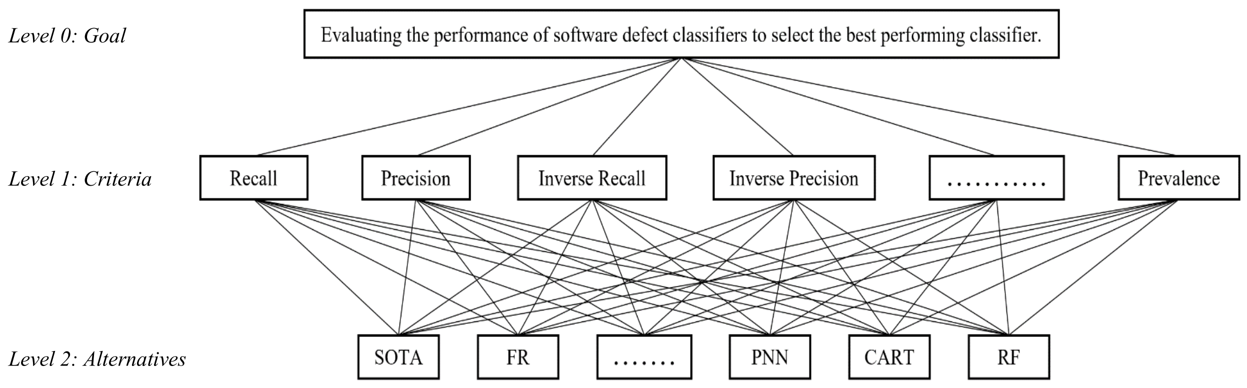

**FIGURE 2.** **FAHP hierarchical structure.**

weights established by the authors judgments for these measures.

By assuming that we have $p$ performance evaluation measures (i.e. criteria), we can construct the criteria pairwise comparison matrix $\tilde{A}$ as follows:

$$\tilde{A}\,[p \times p] = \begin{bmatrix} \tilde{a}_{11} & \cdots & \tilde{a}_{1k} \\ \vdots & \ddots & \vdots \\ \tilde{a}_{j1} & \cdots & \tilde{a}_{jk} \end{bmatrix} \tag{5}$$

where $j = 1 \cdots p \& k = 1 \cdots p$.

Every entry $\tilde{a}_{jk}$ represents the importance of criterion $j$ relative to criterion $k$, where $\tilde{a}_{jk} = (1, 1, 1)\, \forall j = k$.

Once matrix $\tilde{A}$ is constructed, we can calculate the *criteria fuzzy weights vector* $\tilde{\boldsymbol{w}}$ by applying the Geometric Mean method proposed by Buckley [55]. The method can be applied by three steps:

**Firstly**, we calculate the fuzzy geometric mean value $\tilde{r}_j$ for each row $j$ in $\tilde{A}_j$

$$\tilde{r}_j = \left( \left( \prod_{k=1}^{p} l_k \right)^{1/p}, \left( \prod_{k=1}^{p} m_k \right)^{1/p}, \left( \prod_{k=1}^{p} u_k \right)^{1/p} \right) \tag{6}$$

**Secondly**, sum all fuzzy geometric mean values column wise and find their reciprocal, $\left(\tilde{r}_1 \oplus \tilde{r}_2 \oplus \cdots \oplus \tilde{r}_j\right)^{-1}$. The multiplication and addition of two fuzzy numbers operations are defined as,

$$\begin{aligned} \tilde{a}_1 \otimes \tilde{a}_2 &= (l_1, m_1, u_1) \otimes (l_2, m_2, u_2) \\ &= (l_1 \times l_2, m_1 \times m_2, u_1 \times u_2) \qquad (7) \\ \tilde{a}_1 \oplus \tilde{a}_2 &= (l_1, m_1, u_1) \oplus (l_2, m_2, u_2) \\ &= (l_1 + l_2, m_1 + m_2, u_1 + u_2) \qquad (8) \end{aligned}$$

**Lastly**, calculate the *criteria fuzzy weights vector* $\tilde{\boldsymbol{w}}$,

$$\tilde{w}_j = \tilde{r}_j \otimes \left(\tilde{r}_1 \oplus \tilde{r}_2 \oplus \cdots \oplus \tilde{r}_j\right)^{-1} \tag{9}$$

To ease the comparisons of classifiers' rankings, we can defuzzify $\tilde{\boldsymbol{w}}$ using the center of area COA concept [56], Table 7.

$$w_j = \left( \frac{l + m + u}{3} \right), j = 1 \cdots p \tag{10}$$

**Step 3:**

Perform pairwise comparisons between all classifiers with respect to every criterion. At the end of this step, the classifiers scores matrix $S$ is constructed.

$$S\,[c \times p] = \begin{bmatrix} s_{11} & \cdots & s_{1j} \\ \vdots & \ddots & \vdots \\ s_{i1} & \cdots & s_{ij} \end{bmatrix} \tag{11}$$

where $i = 1 \cdots c \,\&\, j = 1 \cdots p$.

Every entry $s_{ij}$ of matrix $S$ represents the score of the $i^{th}$ classifier with respect to the $j^{th}$ criterion. To construct the matrix $S$, we have first to compute classifiers pairwise comparison $B^{(j)}$ matrixes with respect to every criterion $j$.

$$B^{(j)}\,[c \times c] = \begin{bmatrix} b_{11} & \cdots & b_{1h} \\ \vdots & \ddots & \vdots \\ b_{i1} & \cdots & b_{ih} \end{bmatrix} \tag{12}$$

where $i = 1 \cdots c \,\&\, h = 1 \cdots c$.

Each entry $b_{ih}^{(j)}$ of the matrix $B^{(j)}$ represents the evaluation of the $i^{th}$ classifier compared to the $h^{th}$ classifier with respect to the $j^{th}$ criterion. We can compute $b_{ih}^{(j)}$ by dividing the performance evaluation of classifier $i$ over the performance evaluation of classifier $h$. If $b_{ih}^{(j)} > 1$, then the $i^{th}$ classifier is better than the $h^{th}$ classifier, and if $b_{ih}^{(j)} < 1$, then the $i^{th}$ classifier is worse than the $h^{th}$ classifier. When two classifiers performances are equal, then $b_{ih}^{(j)} = 1$. Matrix $B$ entries satisfy the following properties:

$b_{ih}^{(j)} \cdot b_{hi}^{(j)} = 1$ and $b_{ih}^{(j)} = 1, \forall i = h$.

The matrix $E[c \times p]$ entries are utilized in computing $B^{(j)}$ matrixes. The matrix $E$ contains the performance evaluation of each classifier presented by the 22-performance measures. In total we have 12 $E$ matrixes for the 12 data sets experimented. The process of computing $E$ matrixes is as follows:

```
1. Start KNIME
   for each d ∈ D do ▷D is the set of 12 NASA data sets
   and d is a data set
      2. Load data set d
```

**TABLE 6.** The relative fuzzy weights established for the evaluation measures, $\tilde{A}$ matrix.

| | F1 score | AUC ROC | G-Means1 | G-Means2 | MCC | Kappa | Recall | Precision | Inverse Recall | Inverse Precision | FDR | FOR | FPR | FNR | Predicted Positive Condition Rate | ACC | Informedness | Markedness | LR+ | LR− | DOR | Prevalence |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 score | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 | 6,7,8 | 6,7,8 | 6,7,8 | 6,7,8 | 9,9,9 | 9,9,9 | 9,9,9 | 9,9,9 |
| AUC ROC | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 | 6,7,8 | 6,7,8 | 6,7,8 | 6,7,8 | 9,9,9 | 9,9,9 | 9,9,9 | 9,9,9 |
| G-Means1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 | 6,7,8 | 6,7,8 | 6,7,8 | 6,7,8 | 9,9,9 | 9,9,9 | 9,9,9 | 9,9,9 |
| G-Means2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 | 6,7,8 | 6,7,8 | 6,7,8 | 6,7,8 | 9,9,9 | 9,9,9 | 9,9,9 | 9,9,9 |
| Matthews correlation coefficient MCC | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 | 6,7,8 | 6,7,8 | 6,7,8 | 6,7,8 | 9,9,9 | 9,9,9 | 9,9,9 | 9,9,9 |
| Cohen's Kappa | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 | 6,7,8 | 6,7,8 | 6,7,8 | 6,7,8 | 9,9,9 | 9,9,9 | 9,9,9 | 9,9,9 |
| Recall | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 | 6,7,8 | 6,7,8 | 6,7,8 | 6,7,8 |
| Precision | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 | 6,7,8 | 6,7,8 | 6,7,8 | 6,7,8 |
| Inverse Recall | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 | 6,7,8 | 6,7,8 | 6,7,8 | 6,7,8 |
| Inverse Precision | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 | 6,7,8 | 6,7,8 | 6,7,8 | 6,7,8 |
| False Discovery Rate FDR | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/6,1/5,1/7 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 |
| False Omission Rate FOR | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/6,1/5,1/7 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 |
| False Positive Rate FPR | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/6,1/5,1/7 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 |
| False Negative Rate FNR | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/6,1/5,1/7 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 | 4,5,6 | 4,5,6 | 4,5,6 | 4,5,6 |
| Predicted Positive Condition Rate | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 |
| Accuracy ACC | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 |
| Informedness | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 |
| Markedness | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 | 2,3,4 | 2,3,4 | 2,3,4 | 2,3,4 |
| Positive likelihood ratio LR+ | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 |
| Negative likelihood ratio LR− | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 |
| Diagnostic odds ratio DOR | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 |
| Prevalence | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/9,1/9,1/9 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/8,1/7,1/6 | 1/6,1/5,1/4 | 1/6,1/5,1/5 | 1/6,1/5,1/6 | 1/6,1/5,1/7 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1/4,1/3,1/2 | 1,1,1 | 1,1,1 | 1,1,1 | 1,1,1 |

**TABLE 7.** Computing the criteria weights vectors $\tilde{w}$ and $w$.

| | Fuzzy Geometric Mean Value $\tilde{r}$ $(l, u, m)$ | | | Fuzzy Weights $\tilde{w}$ $(l, u, m)$ | | | Defuzzified Weights $w$ | Normalized Defuzzified Weights $w$ |
|---|---|---|---|---|---|---|---|---|
| | ***lower*** **(*l*)** | ***middle*** **(*m*)** | ***upper*** **(*u*)** | ***lower*** **(*l*)** | ***middle*** **(*m*)** | ***upper*** **(*u*)** | | |
| F1 score | 3.014 | 3.475 | 3.878 | 0.076 | 0.102 | 0.135 | 0.105 | 0.101 |
| AUC ROC | 3.014 | 3.475 | 3.878 | 0.076 | 0.102 | 0.135 | 0.105 | 0.101 |
| G-Mean1 | 3.014 | 3.475 | 3.878 | 0.076 | 0.102 | 0.135 | 0.105 | 0.101 |
| G-Mean2 | 3.014 | 3.475 | 3.878 | 0.076 | 0.102 | 0.135 | 0.105 | 0.101 |
| MCC | 3.014 | 3.475 | 3.878 | 0.076 | 0.102 | 0.135 | 0.105 | 0.101 |
| Kappa | 3.014 | 3.475 | 3.878 | 0.076 | 0.102 | 0.135 | 0.105 | 0.101 |
| Recall | 1.385 | 1.727 | 2.153 | 0.035 | 0.051 | 0.075 | 0.054 | 0.052 |
| Precision | 1.385 | 1.727 | 2.153 | 0.035 | 0.051 | 0.075 | 0.054 | 0.052 |
| Inverse Recall | 1.385 | 1.727 | 2.153 | 0.035 | 0.051 | 0.075 | 0.054 | 0.052 |
| Inverse Precision | 1.385 | 1.727 | 2.153 | 0.035 | 0.051 | 0.075 | 0.054 | 0.052 |
| (FDR) | 0.696 | 0.864 | 1.077 | 0.018 | 0.025 | 0.038 | 0.027 | 0.026 |
| (FOR) | 0.696 | 0.864 | 1.077 | 0.018 | 0.025 | 0.038 | 0.027 | 0.026 |
| (FPR) | 0.696 | 0.864 | 1.077 | 0.018 | 0.025 | 0.038 | 0.027 | 0.026 |
| (FNR) | 0.696 | 0.864 | 1.077 | 0.018 | 0.025 | 0.038 | 0.027 | 0.026 |
| Predicted Positive Condition Rate | 0.361 | 0.439 | 0.541 | 0.009 | 0.013 | 0.019 | 0.014 | 0.013 |
| (BACC) | 0.361 | 0.439 | 0.541 | 0.009 | 0.013 | 0.019 | 0.014 | 0.013 |
| Informedness | 0.361 | 0.439 | 0.541 | 0.009 | 0.013 | 0.019 | 0.014 | 0.013 |
| Markedness | 0.361 | 0.439 | 0.541 | 0.009 | 0.013 | 0.019 | 0.014 | 0.013 |
| (LR+) | 0.211 | 0.236 | 0.272 | 0.005 | 0.007 | 0.009 | 0.007 | 0.007 |
| (LR−) | 0.211 | 0.236 | 0.272 | 0.005 | 0.007 | 0.009 | 0.007 | 0.007 |
| (DOR) | 0.211 | 0.236 | 0.272 | 0.005 | 0.007 | 0.009 | 0.007 | 0.007 |
| Prevalence | 0.211 | 0.236 | 0.272 | 0.005 | 0.007 | 0.009 | 0.007 | 0.007 |
| **Sum** | **28.698** | **33.915** | **39.438** | | | | | |
| **Sum$^{-1}$** | **0.025** | **0.029** | **0.035** | | | | | |

```
        3. Run every classifier to generate its confusion matrix
        4. Use the generated confusion matrix to compute
           the 22-performance measures
        5. Construct the corresponding E matrix
    end for
```

Once $B^{(j)}$ matrixes are computed, they need to be normalized column wise. That is, divide each entry $b_{ih}$ in a particular column $h$ over the sum of all entries of this column, Equation (13). This operation is repeated for all columns in matrix $B^{(j)}$.

$$\bar{b}_{ih} = \frac{b_{ih}}{\sum_{i=1}^{c} b_{ih}} \tag{13}$$

We use Equation (14) to find the scores vector $\boldsymbol{s}^{(j)}$ that contains the classifiers' pairwise comparisons scores with respect to every criterion $j$. The c-dimension column vector $\boldsymbol{s}^{(j)}$ is

**TABLE 8.** Classifiers ranks per every data sets.

| | Data sets | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Ranks** | CM1 | JM1 | KC1 | KC3 | MC1 | MC2 | MW1 | PC1 | PC2 | PC3 | PC4 | PC5 |
| **1** | MLP | ***RF*** | ***RF*** | CART | ***RF*** | MLP | ***RF*** | FR | FR | ***RF*** | ***RF*** | ***RF*** |
| **2** | CART | FR | SOTA | DT | FR | NB | FR | ***RF*** | DT | FR | FR | FR |
| **3** | FR | CART | CART | FR | DT | ***RF*** | DT | SOTA | ***RF*** | CART | MLP | DT |
| **4** | DT | DT | DT | PNN | KNN | FR | CART | CART | MLP | DT | CART | CART |
| **5** | ***RF*** | KNN | FR | MLP | CART | KNN | MLP | DT | KNN | SOTA | KNN | MLP |
| **6** | SOTA | SOTA | KNN | ***RF*** | SOTA | CART | LR | KNN | SOTA | KNN | PNN | PNN |
| **7** | LR | MLP | PNN | KNN | MLP | LR | SOTA | MLP | CART | MLP | DT | KNN |
| **8** | KNN | LR | MLP | SOTA | PNN | PNN | PNN | PNN | LR | PNN | SOTA | SOTA |
| **9** | PNN | PNN | LR | NB | LR | SOTA | KNN | LR | PNN | LR | LR | LR |
| **10** | NB | NB | NB | LR | NB | DT | NB | NB | NB | NB | NB | NB |
| **11** | SVM | SVM | SVM | SVM | SVM | SVM | SVM | SVM | SVM | SVM | SVM | SVM |

computed by taking the averages row-wise for every row $i$ in $B^{(j)}$.

$$s^{(j)} = \frac{\sum_{h=1}^{c} \bar{b}_{ih}}{c} \tag{14}$$

Now, we can construct matrix $S$ by combining all calculated $\boldsymbol{s}^{(j)}$ scores vectors,

$$S = \left[\boldsymbol{s}^{(1)} \cdots \boldsymbol{s}^{(j)}\right], \quad where\ j = 1 \cdots p \tag{15}$$

Each column in the matrix $S$ corresponds to one of the $\boldsymbol{s}^{(j)}$ column vectors.

**Step 4:**

Calculate the vector $\boldsymbol{v}$ of the classifiers' priorities by multiplying the classifiers' pairwise comparison scores matrix $S$ by the defuzzified *criteria weights vector* $\boldsymbol{w}$, Equation (16).

$$\boldsymbol{v} = S \cdot \boldsymbol{w} \tag{16}$$

Each $\boldsymbol{v}_i$ entry represents the score (i.e. rank) assigned by the FAHP process to the $i^{th}$ classifier in comparison to all other $c - 1$ classifiers.

## VIII. RESULT

The experiments resulted in 12 $E$ matrixes, 12 $S$ matrixes, and 264 $B$ matrixes. For brevity, we will report the summary of the results.

We can notice from Table 8 that every data set reveals a unique order of the experimented classifiers performance ranks. These results conform with much published research that every data set (i.e. software project) is a unique product and persists unique characteristics. Kastro and Bener [57] concluded in their study that it is almost impossible to have two identical software products in terms of developing process, programming languages used, programmers' experience, algorithm complexity, or even the development methodology. Myrtveit *et al.* [8] reported similar findings.

However, some interesting trends can be implied. Random Forest ***(RF)*** has won the first rank 7 times and the second rank one time. Fuzzy Rule (FR) has won the first rank 2 times and the second rank 6 times. This shows that those particular classifiers perform very well. On the contrary, SVM has won the last rank (i.e. the 11th rank) 12 times, which implies that this classifier consistently performs poorly in these experiments. Close to this performance is NB that won the 10th rank 10 times, the 9th rank 1 and surprisingly won the 2nd rank 1 time.

To make clearer final comparisons between all competing classifiers, Table 9 shows the average rank for each classifier over all experimented data sets. The procedure we follow is to count the number of times each classifier achieves a particular rank, then multiply this number by the rank itself. The sum of these numbers is divided over the total number of available ranks. Small average rank values indicate better performing classifiers in comparison to classifiers that have larger averages. Table 9 confirms our earlier observations in this section.

On the other hand, we averaged the matrix $E$ for the 12 data sets and applied FAHP on this one averaged matrix. As expected, the final rankings are perfectly matching the previous ones.

## IX. THREATS TO VALIDIT

The first threat to validity comes from the fact that this paper's results and conclusions are biased based on the data sets and classifiers we used [58]. However, we believe that by choosing the publicly available NASA data sets, replication should be possible and encouraged by other researchers. The same argument applies for choosing the most common classifiers in the field of software defect prediction [4], [38], [57]. Moreover, NASA data sets meet all the requirements that would increase our research external validity stated by Khoshgoftaar *et al.* [59], that is, increasing the generalization of the results outside our experimental settings:

1) Be large enough to be comparable to real industry projects
2) Developed in an industrial environment, rather than an artificial setting
3) Developed by a group of developers rather than an individual
4) Developed by professionals, rather than students

On the other hand, and in order to decrease the presence of internal validity threats, we decided to use the cleaned-up

**TABLE 9. Averaged data sets ranks.**

| | Classifiers | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Ranks** | RF | FR | CART | DT | MLP | KNN | SOTA | PNN | LR | NB | SVM |
| **1** | 7 | 2 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 2 | 12 | 2 | 4 | 0 | 0 | 2 | 0 | 0 | 2 | 0 |
| **3** | 6 | 6 | 9 | 9 | 3 | 0 | 3 | 0 | 0 | 0 | 0 |
| **4** | 0 | 4 | 16 | 16 | 4 | 4 | 0 | 4 | 0 | 0 | 0 |
| **5** | 5 | 5 | 5 | 5 | 15 | 20 | 5 | 0 | 0 | 0 | 0 |
| **6** | 6 | 0 | 6 | 0 | 0 | 18 | 24 | 12 | 6 | 0 | 0 |
| **7** | 0 | 0 | 7 | 7 | 28 | 14 | 7 | 7 | 14 | 0 | 0 |
| **8** | 0 | 0 | 0 | 0 | 8 | 8 | 24 | 40 | 16 | 0 | 0 |
| **9** | 0 | 0 | 0 | 0 | 0 | 9 | 9 | 27 | 54 | 9 | 0 |
| **10** | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 10 | 100 | 0 |
| **11** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 132 |
| | | | | | | | | | | | |
| **Sum** | 26 | 29 | 46 | 51 | 60 | 73 | 74 | 90 | 100 | 111 | 132 |
| **Average** | 2.4 | 2.6 | 4.2 | 4.6 | 5.5 | 6.6 | 6.7 | 8.2 | 9.1 | 10.1 | 12.0 |

NASA date sets instead of the original ones, as discussed earlier in the data sets section. This allows us to avoid the noise sources that exist in the original NASA data sets.

Moreover, some data sets contain relatively small number of modules, such as, MC2 and KC3. Especially when 10-fold cross-validation technique is employed. Some classifiers that are sensitive to the size of data sets might lose some of their performance quality [60]. This effect might be increased after performing the cleaning procedures on NASA data sets. As Table 3 shows, this resulted in smaller number of observations for each experimented data set.

## X. CONCLUSIONS

There is a substantial need to design and develop reliable software defect classifiers that classify software components into defective and non-defective. The benefit of this objective is the ability to focus software defect detection efforts and project resources on part of a system rather than testing the whole system.

Nevertheless, the major problem that software practitioners face is how to reliably evaluate classifiers and how to select the best fit for their software development projects. Since the evaluation of software defect classifiers performance is highly dependent on the specific measures employed, the performance evaluation might improve or deteriorate if practitioners choose different performance measures.

As we believe that performance evaluation must be seen as a comprehensive strategy rather than relying on preferably selected performance measures, Fuzzy Analytical Hierarchy Process (FAHP) is implemented in this research to satisfy this requirement. FAHP allowed us to combine a wider spectrum of evaluation measures, in contrast to relying on preferably selected one or two evaluation measures. Another strength comes from the fact that FAHP employs fuzzy membership function to account for human nature of uncertainty and vagueness when evaluating and comparing performance measures against each other. The results show that this approach will increase software developers' confidence in research outcomes and help them in avoiding false conclusions and infer reasonable boundaries for them.

## REFERENCES

[1] B. Turhan, A. Tosun, and A. Bener, "Empirical evaluation of mixed-project defect prediction models," in *Proc. 37th EUROMICRO Conf. Softw. Eng. Adv. Appl.*, Oulu, Finland, Aug. 2011, pp. 396–403.

[2] D. Lo, S.-C. Khoo, J. Han, and C. Liu, Eds., *Mining Software Specifications: Methodologies and Applications*. Boca Raton, FL, USA: CRC Press, 2011, pp. 1–15.

[3] M. Shepperd, D. Bowes, and T. Hall, "Researcher bias: The use of machine learning in software defect prediction," *IEEE Trans. Softw. Eng.*, vol. 40, no. 6, pp. 603–616, Jun. 2014.

[4] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 485–496, Jul. 2008.

[5] L. Madeyski and M. Jureczko, "Which process metrics can significantly improve defect prediction models? An empirical study," *Softw. Qual. J.*, vol. 23, no. 3, pp. 393–422, 2014.

[6] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: A benchmark and an extensive comparison," *Empir. Softw. Eng.*, vol. 17, nos. 4–5, pp. 531–577, 2011.

[7] R. S. Wahono, N. S. Herman, and S. Ahmad, "A comparison framework of classification models for software defect prediction," *Adv. Sci. Lett.*, vol. 20, nos. 10–11, pp. 1945–1950, 2014.

[8] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and validity in comparative studies of software prediction models," *IEEE Trans. Softw. Eng.*, vol. 31, no. 5, pp. 380–391, May 2005.

[9] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1276–1304, Nov. 2012.

[10] H. Wang, T. M. Khoshgoftaar, and Q. Liang, "A study of software metric selection techniques: Stability analysis and defect prediction model performance," *Int. J. Artif. Intell. Tools*, vol. 22, no. 5, 2013, Art. no. 1360010.

[11] I. Myrtveit and E. Stensrud, "Validity and reliability of evaluation procedures in comparative studies of effort prediction models," *Empirical Softw. Eng.*, vol. 17, nos. 1–2, pp. 23–33, 2012.

[12] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.

[13] N. Fenton and B. Kitchenham, "Validating software measures," *Softw. Test., Verification Rel.*, vol. 1, no. 2, pp. 27–42, 1991.

[14] C. Andersson, "A replicated empirical study of a selection method for software reliability growth models," *Empirical Softw. Eng.*, vol. 12, no. 2, p. 161, 2007.

[15] L. J. White, "The importance of empirical work for software engineering papers," *Softw. Test., Verification Rel.*, vol. 12, no. 4, pp. 195–196, 2002.

[16] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement," *ACM SIGKDD Explorations Newslett.*, vol. 12, no. 1, pp. 49–57, 2010.

[17] D. C. Ince, L. Hatton, and J. Graham-Cumming, "The case for open computer programs," *Nature*, vol. 482, no. 7386, p. 485, 2012.

[18] B. Kitchenham, "What's up with software metrics?—A preliminary mapping study," *J. Syst. softw.*, vol. 83, no. 1, pp. 37–51, 2010.

[19] N. Fenton and J. Bieman, *Software Metrics: A Rigorous and Practical Approach*, 3rd ed. Boca Raton, FL, USA: CRC Press, 2014.

[20] B. D. Rouyendegh, "The DEA and intuitionistic fuzzy TOPSIS approach to departments' performances: A pilot study," *J. Appl. Math.*, vol. 2011, Oct. 2011, Art. no. 712194. doi: 10.1155/2011/712194.

[21] T. L. Saaty, "Decision making with the analytic hierarchy process," *Int. J. Services Sci.*, vol. 1, no. 1, pp. 83–98, 2008.

[22] T. L. Saaty, *Analytical Hierarchy Process*. New York, NY, USA: McGraw-Hill, 1980.

[23] S. Kubler, J. Robert, W. Derigent, A. Voisin, and Y. Le Traon, "A state-of the-art survey & testbed of fuzzy AHP (FAHP) applications," *Expert Syst. Appl.*, vol. 65, pp. 398–422, Dec. 2016.

[24] B. D. Rouyendegh and T. Erkart, "Selection of academic staff using the fuzzy analytic hierarchy process (FAHP): A pilot study," *Tehnicki Vjesnik*, vol. 19, no. 4, pp. 923–929, 2012.

[25] M. Z. Naghadehi, R. Mikaeil, and M. Ataei, "The application of fuzzy analytic hierarchy process (FAHP) approach to selection of optimum underground mining method for Jajarm Bauxite Mine, Iran," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 8218–8226, 2009.

[26] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, Jun. 1965.

[27] Y. Peng, G. Kou, G. Wang, W. Wu, and Y. Shi, "Ensemble of software defect predictors: An AHP-based evaluation method," *Int. J. Inf. Technol. Decis. Making*, vol. 10, no. 1, pp. 187–206, 2011.

[28] G. Kabir and M. A. A. Hasin, "Comparative analysis of AHP and fuzzy AHP models for multicriteria inventory classification," *Int. J. Fuzzy Logic Syst.*, vol. 1, no. 1, pp. 1–16, 2011.

[29] P. J. M. van Laarhoven and W. Pedrycz, "A fuzzy extension of Saaty's priority theory," *Fuzzy Sets Syst.*, vol. 11, nos. 1–3, pp. 229–241, 1983.

[30] J. Harding, E. A. Walker, and C. L. Walker, *The Truth Value Algebra of Type-2 Fuzzy Sets: Order Convolutions of Functions on the Unit Interval*. Boca Raton, FL, USA: CRC Press, 2016.

[31] Y. Jiang, B. Cukic, and Y. Ma, "Techniques for evaluating fault prediction models," *Empirical Softw. Eng.*, vol. 13, no. 5, pp. 561–595, 2008.

[32] M. Vihinen, "How to evaluate performance of prediction methods? Measures and their interpretation in variation effect analysis," *BMC Genomics*, vol. 13, no. 4, p. S2, 2012. doi: 10.1186/1471-2164-13-S4-S2.

[33] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification," *Pattern Recognit. Lett.*, vol. 30, no. 1, pp. 27–38, 2009.

[34] M. Shepperd, Q. Song, Z. Sun, and C. Mair. *NASA MDP Software: Defects Datasets*. Accessed: Sep. 16, 2018. [Online]. Available: https://figshare.com

[35] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Trans. Rel.*, vol. 62, no. 2, pp. 434–443, Jun. 2013.

[36] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the nasa software defect datasets," *IEEE Trans. Softw. Eng.*, vol. 39, no. 9, pp. 1208–1215, Sep. 2013. doi: 10.1109/TSE.2013.11.

[37] W. Fan, F. Geerts, and X. Jia, "A revival of integrity constraints for data cleaning," *Proc. VLDB Endowment*, vol. 1, no. 2, pp. 1522–1523, 2008.

[38] T. M. Khoshgoftaar, K. Gao, A. Napolitano, and R. Wald, "A comparative study of iterative and non-iterative feature selection techniques for software defect prediction," *Inf. Syst. Frontiers*, vol. 16, no. 5, pp. 801–822, 2014.

[39] N. V. Nenkov and I. Ibryam, "A survey of the open source platforms Rapidminer and Konstanz Information Miner (KNIME) for data processing, analysis and mining," Pedagogical College, Dobrich, Bulgaria, Tech. Rep., 2013, pp. 124–129.

[40] M. R. Berthold *et al.*, "KNIME-the Konstanz information miner: Version 2.0 and beyond," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 26–31, 2009.

[41] M. R. Berthold, "A probabilistic extension for the DDA algorithm," in *Proc. IEEE Int. Conf. Neural Netw.*, Washington, DC, USA, vol. 1, Jun. 1996, pp. 341–346.

[42] J. Herrero, A. Valencia, and J. Dopazo, "A hierarchical unsupervised growing neural network for clustering gene expression patterns," *Bioinformatics*, vol. 17, no. 2, pp. 126–136, 2001.

[43] H. Enderton and H. B. Enderton, *A Mathematical Introduction to Logic*. Amsterdam, The Netherlands: Elsevier, 2001.

[44] D. R. Cox, "The regression analysis of binary sequences," *J. Roy. Stat. Soc., B (Methodol.)*, vol. 20, no. 2, pp. 215–232, 1958.

[45] R. Stuart and N. Peter, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, 2003.

[46] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: Nn Pattern Classification Techniques*. Washington, DC, USA: IEEE Computer Society Press, 1991.

[47] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE Int. Conf. Neural Netw.*, San Francisco, CA, USA, Mar. 1993, pp. 586–591.

[48] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods*. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.

[49] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Comput.*, vol. 13, no. 3, pp. 637–649, 2001.

[50] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Amsterdam, The Netherlands: Elsevier, 2014.

[51] J. Shafer, R. Agrawal, and M. Mehta, "SPRINT: A scalable parallel classifier for data mining," in *Proc. Int. Conf. Very Large Data Bases*, Bombay, India, 1996, pp. 544–555.

[52] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees* (The Wadsworth Statistics and Probability Series). Belmont, CA, USA: Wadsworth International Group, 1984, p. 356.

[53] W.-Y. Loh, "Classification and regression trees," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[54] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction," in *Proc. 18th Int. Conf. Eval. Assessment Softw. Eng.*, London, U.K., 2014, p. 43.

[55] J. J. Buckley, "Fuzzy hierarchical analysis," *Fuzzy Sets Syst.*, vol. 17, pp. 233–247, Dec. 1985.

[56] B. Schott and T. Whalen, "Nonmonotonicity and discretization error in fuzzy rule-based control using COA and MOM defuzzification," in *Proc. IEEE 5th Int. Fuzzy Syst.*, New Orleans, LA, USA, vol. 1, Sep. 1996, pp. 450–456.

[57] Y. Kastro and A. B. Bener, "A defect prediction method for software versioning," *Softw. Qual. J.*, vol. 16, no. 4, pp. 543–562, 2008.

[58] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–13, Jan. 2007.

[59] T. M. Khoshgoftaar, N. Seliya, and N. Sundaresh, "An empirical study of predicting software faults with case-based reasoning," *Softw. Quality J.*, vol. 14, no. 2, pp. 85–111, 2006.

[60] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Inf. Sci.*, vol. 179, no. 8, pp. 1040–1058, 2009.

**HUSSAM GHUNAIM** received the B.Sc. degree in physics and mathematics, in 1993, and the M.Sc. degree (Hons.) in Internet engineering from London South Bank University, U.K., in 2005. He is currently pursuing the Ph.D. degree in computer science and engineering with the University of Bridgeport, Bridgeport, CT, USA. He was engaged in progressive career in education for over 20 years. He is an active Researcher in software defect prediction modeling techniques and performance evaluation procedures.

**JULIUS DICHTER** received the B.S. degree from the University of Connecticut, the M.S. degree from the University of New Haven, and the Ph.D. degree from the University of Connecticut. He is currently an Associate Professor of computer science with the University of Bridgeport, Bridgeport, CT, USA. His research interest includes the optimization of parallel computing. He is a member of the IEEE Computer Society, ACM, International Society of Computers and Their Applications, and UPE Computer Science Honor Society.