IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 19, 2019, accepted June 4, 2019, date of publication June 19, 2019, date of current version July 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2923821

Analyses of Classifier's Performance Measures Used in Software Fault Prediction Studies

MUHAMMAD RIZWAN[1], AAMER NADEEM[1], AND MUDDASSAR AZAM SINDHU[2]
[1]Department of Computer Science, Capital University of Science and Technology (CUST), Islamabad 45750, Pakistan
[2]Department of Computer Science, Quaid-i-Azam University (QAU), Islamabad 45320, Pakistan

Corresponding author: Muhammad Rizwan (rizwanabuahmad@gmail.com)

ABSTRACT Assessing the quality of the software is both important and difficult. For this purpose, software fault prediction (SFP) models have been extensively used. However, selecting the right model and declaring the best out of multiple models are dependent on the performance measures. We analyze 14 frequently used, non-graphic classifier's performance measures used in SFP studies. These analyses would help machine learning practitioners and researchers in SFP to select the most appropriate performance measure for the models' evaluation. We analyze the performance measures for resilience against producing invalid values through our proposed plausibility criterion. After that, consistency and discriminancy analyses are performed to find the best out of the 14 performance measures. Finally, we draw the order of the selected performance measures from better to worse in both balance and imbalance datasets. Our analyses conclude that the F-measure and the G-mean1 are equally the best candidates to evaluate the SFP models with careful analysis of the result, as there is a risk of invalid values in certain scenarios.

INDEX TERMS Classification, evaluation parameters, machine learning, performance measures, software fault prediction.

I. INTRODUCTION

Software fault prediction (SFP) is a means to detect fault-prone components or a number of expected faults in a software component. It is generally, done by using the dataset from the prior releases of the same software system or different software, as is done in Cross Product Defect Prediction (CPDP) [1]. The dataset is used in model building, and then predicting faults in the system under development/test. SFP helps in reducing testing cost and improving the quality of the system. Moreover, this can direct the testing team to focus more on the fault-prone modules. Predicting the number of faults potentially provides the criteria for stopping of the testing process.

SFP is usually done through Machine learning (ML) algorithms, statistical algorithms or expert's opinion [2]. ML, besides being the most used model building technique [3], significantly improves classification accuracy [4]. Numerous ML models are being used in SFP. Performance of these ML models has been compared in [5]. In which authors reported that Decision Tree and Naïve Bayes are the most used ML algorithms in SFP studies.

The associate editor coordinating the review of this manuscript and approving it for publication was Mehul S. Raval.

The declaration of a model as *the best* is dependent on the performance measure [6] or sometimes multiple performance measures [7]. Therefore, selection of performance measures while keeping in view their scope, relationship and interpretation are of key importance, which is the primary focus of this paper. The task is even more important when two performance measures consent on two classifiers' performance on one test set, may conflict on some other test set.

In the classification, performance measures are derived out of a confusion matrix, which is a useful tool for analyzing the goodness of a classifier. Existing studies to evaluate these measures either do not address the performance measures used in SFP exclusively or limited to very few SFP specific performance measures. Moreover, there is a sporadic address on the recommendation of the performance measure. For these reasons, the objectives of this paper include:

1) A survey of commonly used performance measures in SFP.
2) Identification of comparative techniques used to evaluate performance measures.
3) A comparison of the surveyed performance measures and guide to their merits and demerits.

In this paper, we list out 14 performance measures which are commonly used in SFP studies. The performance measures

TABLE 1. Studies on performance measures and their respective evaluation parameters.

Studies	Evaluated performance measures	Comparative techniques
[11]	Accuracy, AUC	ANOVA and Duncan's multiple range test
[12]	Accuracy, AUC, TPR, Precision, F-measure, G-mean2	Theoretical discussion
[13]	Accuracy, Precision, F-measure, WRAcc (Weighted Relative Accuracy)	ROC Isometrics
[14]	AUC, Error rate	Statistical analysis (Mean, relative standard deviation etc.)
[15]	AUC, Error rate	Stability, Discriminancy, "bias-variance" trade-off
[16]	Accuracy, Lift, F-Score, AUC, Average Precision, Precision/Recall Break-Even Point, Squared Error, Cross Entropy, and Probability Calibration	Multidimensional scaling
[17]	Accuracy, weighted relative Accuracy, entropy, Gini index, Precision, AUC	Isometrics and equivalence
[18]	AUC, Accuracy	Strict/statistical consistency and Discriminancy
[19]	Precision-recall curve and ROC curve	Curve dominate theorem
[20]	AUC, Accuracy, RMS, AUC:acc	Degree of consistency and discriminancy
[21]	Accuracy, TPR, Precision, Type-I error, Type II error	Faults coverage
[22]	Accuracy, Error rate, Sensitivity, Specificity, Precision, G-mean, F-measure and *J* coefficient, ROC curve, Precision, and Recall curve, Cost Curve, and Lift Chart	Friedman test, Nemenyi test
[23]	Accuracy, Kappa statistic, F-measure, Macro average arithmetic, Macro average geometric, AUNU, AUNP, AU1UAU1P, AU1P, SAUC, PAUC, MAPR, MPR, MAE, MSE, LogL, CalL, CalB	Statistical correlation, Misclassification noise. Probability noise, Ranking noise, Class proportion noise
[24]	TPR, TNR, Accuracy, Precision, false discovery rate, *J* coefficient, F1-measure, balanced Accuracy, MCC, FPR, AUC, aiming, coverage, absolute-true-rate, and absolute- false-rate (AFR)	Theoretical discussion

are evaluated through three evaluation measures Plausibility, Consistency, and Discriminancy. In the end, we conclude that F-measure and G-mean1 are better performance measures. Finally, we ordered all the performance measures as per their goodness in the balanced and imbalanced datasets. Our work is significant in the following ways;

1) We discuss as many as 14 performance measures being used in SFP domain.
2) Present a new tool for the evaluation of performance measures i.e. Plausibility.
3) Ordered list of performance measures as per their effectiveness in the balanced and imbalanced datasets.

Rest of the paper is further organized as follows: *Section II* provides the literature review of this field. *Section III* provides the overview of the performance measures used in SFP studies. *Section IV* briefly classifies the performance measures into two classes. *Section* V elaborates the evaluation parameters used to evaluate the performance measures followed by *Section VI*, in which actual evaluation of performance measures is conducted. Comparison of our findings with the existing work is drawn in *Section VII* followed by the conclusion in *Section VIII* . Finally, future directions of the research are given in *Section IX*.

II. RELATED WORK

Literature is quite rich in addressing the SFP [2], [5], [8], [9]. However, evaluation of performance measures, which are used in SFP have attracted sporadic attention (see Table 1).

Bradle compares Area under the Curve (AUC) with Accuracy [10]. He computes the sensitivity of both measures in Analysis of Variance (ANOVA) and Duncan's multiple range test and recommends AUC to be used in SFP studies. Moreover, he observes the effect of increasing samples on standard error. On the other hand, regardless of SFP, Kubat *et al.* discuss the selection of performance measure in satellite image recognition [11]. They go through the theoretical discussion on Accuracy, TPR, Precision, G-mean1, and F-measure. Ultimately, they select Receiver operator curve (ROC) for their experiment.

Peter compares Accuracy, Precision, and F-measure along with several decision tree splitting criteria through isometric plots [12]. He identifies Accuracy as skew sensitive, whereas Precision as an effective skew ratio. Cortes *et al.* conduct a detailed statistical analysis of the relationship between Error rate and AUC [13]. They report the usefulness of an algorithm specifically designed to globally optimize the AUC. Rosset evaluates the AUC and Error rate for discrimination [14]. He concludes the preferability of AUC for being more stable and discriminating. Caruana *et al.* compare nine performance measures including average Precision, Precision/recall, break-even point, squared error etc. without being specific to SFP [15]. Fürnkranz *et al.* declare the equivalence of Precision and cost-weighted difference between positive and negative with conventional measures like Accuracy, weighted relative Accuracy, entropy, and Gini index [16]. A theoretical and empirical comparison between AUC and Accuracy are done in [17]. The study proves the domination of AUC over Accuracy. Davis *et al.* theoretically compare the relationship between Precision-recall and ROC curves [18]. Huang *et al.* use consistency and discriminancy measures for comparing their proposed measure with root mean square and AUC [19]. Jiang *et al.* discuss numerous performance measures, but evaluation is missing [21]. The most comprehensive study was conducted in [22], in which 18 different performance measures are evaluated empirically in several scenarios without limiting to SFP domain. The study

concludes the presence of clustering and the relationship between the measures. Four traits of performance measures are taken; class threshold choice optimality, separability/ranking quality, calibration performance, and sensitivity (or conversely robustness) to changes in prior class distribution. Though the study focus on the classification measure, yet does not address the key performance measures like TPR, Precision etc. Jiao *et al.* focus on performance measures used in bioinformatics [23]. They discuss general purpose of the performance measures without going into the detail of selection and evaluation of the discussed measures. Besides them, few more studies have been conducted in this direction [7], [20].

All of these works either do not address the performance measures in SFP exclusively or limited to very few performance measures used in SFP. Moreover, these works are difficult to compare and understand together because of lack of common domain, common performance measures, and common evaluation methodology. Our work is focused instead only on those performance measures, which are used in SFP studies. Thereafter, we compare as many as 14 performance measures against three evaluation criteria.

III. PERFORMANCE MEASURES USED IN SFP

The performance measures for classifiers are derived out of four cases; these are [1]; true positive (*TP*), false negative (*FN*), true negative (*TN*), and false positive (*FP*). Total positive instances are denoted by *P* and total negative instances are denoted by *N*. In SFP domain, a positive class is a faulty class. In multiclass dataset wherein classes can represent type of fault like Functionality faults, communication faults, syntactic faults etc. or severity of fault [24], [25], like Low, Medium, High. In such a case, the positive class would be the class of interest and the four cases can easily be identified through ''one vs. all'' framework [26]. The selected performance measures are computed once the positive class is identified. In this paper the objective and findings will prevail irrespective of the problem type (binary or multiclass). For simplicity, we discuss only the binary classes. So, if a classifier declares any faulty instance as faulty, the classification is *TP*. If a classifier declares any instance as fault-free when it is actually faulty, the classification is *FN*. If the classifier declares any fault-free instance as fault-free, the classification is *TN*. Finally, if a classifier labels any instance as faulty when it is actually fault-free, the classification is *FP*.

This section briefly describes 14 classifiers' performance measures which are used in SFP (Table 2). However, following classes of measures are out of our paper's scope:

1) Graphical evaluation methods [21] like ROC, Precision-Recall curve, lift chart [18] etc.
2) Least commonly used performance measures in SFP like Effort [27], Efficiency [28], Inspection [29] etc.
3) Performance measures not used in SFP studies like Conditional log likelihood (CLL) [30] etc.

[1] In this paper, the term ''four cases'' would refer *TP*, *FN*, *TN*, and *FP*.

TABLE 2. Usage of performance measures in the SFP studies.

Performance Measure	SFP Studies
Precision	[1, 6, 25, 27, 32–34, 36, 41–56]
TPR	[1, 6, 27, 28, 32, 36, 37, 39, 41–48, 51, 53, 55–70]
TNR	[27, 28, 32, 36, 39, 42, 47, 51, 55–58, 66]
Accuracy	[1, 6, 27, 28, 36, 39, 42, 43, 46, 48, 49, 53, 55, 56, 58, 59, 64, 66, 70–80]
F-measure	[42, 43, 45, 46, 55, 56, 81–83]
G-mean1	[56, 82, 83]
G-mean2	[42, 48, 56, 82, 83]
J coefficient	[56, 84–86]
FPR	[41, 51, 53, 55, 56, 58–70, 87, 88]
FNR	[51, 56, 87, 88]
Type-I error	[6, 29, 34, 75, 89–101]
Type-II error	[6, 29, 34, 75, 89–101]
Error rate	[29, 48, 56, 87–93, 95, 97, 98, 101–104]
Balance	[56, 62, 65, 66, 68, 105]

A. PRECISION

Precision and True positive rate (discussed afterward) typically used in document retrieval, proposed in [31]. Precision is about being precise. Precision measures the chance of correctly predicting faulty modules among the modules classified as fault-prone. It is also named as correctness [32]–[34] and in the field of biomedicine, it is named as positive predictive value (PPV) [35]. Mathematically,

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

B. TRUE POSITIVE RATE (TPR)

TPR measures the ability of a classifier in identifying positive samples. It has some other names also, like recall [36], sensitivity [27], hit rate [37], and probability of detection (PD) [38]. Mathematically, it can be written as;

$$TPR = \frac{TP}{P} \tag{2}$$

Precision-recall curve [18] is another important measure which is a composite of Precision (discussed earlier) and TPR. However, it is beyond the scope of this paper.

C. TRUE NEGATIVE RATE (TNR)

TNR measures the ability of a predictor in identifying negative samples. It is also named as specificity [32], [39]. ROC incorporates TPR (just discussed) and TNR to compute AUC. Mathematically it can be written as;

$$TNR = \frac{TN}{N} \tag{3}$$

D. ACCURACY

Accuracy shows the correct predictions. It is a good measure when the classes in the test dataset are nearly balanced [20]. It measures the ability of a classifier in correctly identifying

all samples, no matter it is positive or negative.

$$Accuracy = \frac{TP + TN}{P + N} \tag{4}$$

E. F-MEASURE

F-measure is the harmonic mean between Precision and TPR, first introduced in [40]. It tells how precise a classifier is (how many instances it classifies correctly), as well as how robust it is. Mathematically it is written as.

$$F\text{-}measure = \frac{(\beta^2 + 1) \times Precision \times TPR}{\beta^2 \times Precision + TPR} \tag{5}$$

where, β can be 0.5, 1 and 2 for F0.5, F1, and F2 measures respectively. Varying the value of β allows different weights to be assigned to Precision and TPR. Detail derivation of F-measure and assigning the value of β are discussed in [35].

F. G-MEAN1 AND G-MEAN2

Geometric mean (G-mean) is proposed in [11]. It has two variants G-mean1 and G-mean2. These measures evaluate the degree of inductive bias in terms of a ratio of positive accuracy and negative accuracy [48].

$$G\text{-}mean1 = \sqrt{TPR \times Precision} \tag{6}$$

$$G\text{-}mean2 = \sqrt{TPR \times TNR} \tag{7}$$

G. J COEFFICIENT

Youden proposes J coefficient back in 1950 [106]. It is also known as Youden' index [107], denoted by γ. J is the Jaccard index. Its value ranges [-1, 1]. Mathematically, it can be written as;

$$J\ coefficient = TPR + TNR - 1 \tag{8}$$

J coefficient equals to one, represents the perfect classification, whereas negative-one shows the worst case.

H. FALSE POSITIVE RATE (FPR) AND FALSE NEGATIVE RATE (FNR)

False positive rate is also called as probability of false detection (PF) [63], [65].

$$FPR = \frac{FP}{FP + TN} \tag{9}$$

FNR computes the degree of false detection of faulty modules;

$$FNR = \frac{FN}{FN + TP} \tag{10}$$

I. TYPE-I ERROR, TYPE-II ERROR AND ERROR RATE

Mathematical form of these measures are;

$$Type\text{-}I\ error = \frac{FP}{P + N} \tag{11}$$

$$Type\text{-}II\ error = \frac{FN}{P + N} \tag{12}$$

$$Error\ rate = \frac{FP + FN}{P + N} \tag{13}$$

Type-I and Type-II errors are also named as Type-I and Type-II misclassifications [20]. Likewise, Error rate is also known as overall misclassification rate [7], [34] or simply, misclassification rate [108].

J. BALANCE

In practice, FPR and TNR (recall) need to be balanced, which is achieved by computing the difference between the best FPR which is zero and the difference between the best TNR which is 1. Rest of the factors are for normalization and computing percentage [61].

$$Balance = 1 - \sqrt{\frac{(0 - FPR)^2 + (1 - TPR)^2}{2}} \tag{14}$$

IV. CLASSIFICATION OF PERFORMANCE MEASURES

The measures discussed in the last section can be classified into two orthogonal classes; (1) Positive and negative oriented measures, (2) base, derived and complement measures[2]. These two classes are shortly discussed below;

A. POSITIVE AND NEGATIVE ORIENTED MEASURES

Positive oriented measures refer to the performance measure whose higher value is desirable. Among the list of measures discussed earlier, Precision, TPR, TNR, Accuracy, F-measure, G-mean1, G-mean2, and J coefficient are positive oriented measures. In contrast to that, where lower values are desirable, are referred to as negative oriented measures. FPR, FNR, Type-I error, Type-II error, Error rate, and Balance are all negative oriented measures.

B. BASE, DERIVED, AND COMPLEMENT MEASURES

As the name implies, base measures are directly computed out of the confusion matrix, whereas derived measures can be computed from base measures. TPR, TNR, Type-I error, Type-II error are all base measures whereas the rest of the measures can be derived out of any or some of these base measures. There can also be a unique category of derived measures i.e. complement measures. These measures have a strict inverse proportionality relationship with some other measures. Three measures which have a complement relationship with the other three measures are;

1) Accuracy to Error rate
2) TPR to FNR
3) TNR to FPR

"Complement" is a symmetric relationship whereas "base and derived" is asymmetric in nature. Figure 1 shows the base, derived, and complement measures. Base measures[3] are filled-in with a black color to discriminate them from derived measures, which are shown by hollow rectangles. Unidirectional arrow is directed from base to derived measures, whereas the bidirectional arrow is between complement measures.

[2]The word "complement" is borrowed from set theory.

[3]Base measures having complement measures can also be taken as base measures.

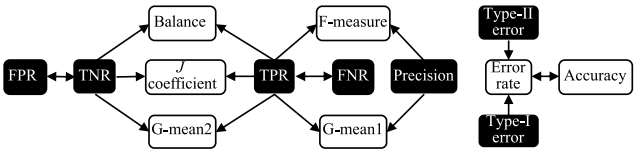


FIGURE 1. Base, derived, and complement measures.

V. EVALUATION CRITERIA

A. PLAUSIBILITY

Plausibility refers to the absence of implausible values. Implausible values can be of three types:

Type:1 Occurrence of "divided by zero". For instance, for (0, 0, 1, 0)[4] F-measure will produce "divided by zero" error. Thus, F-measure will be declared as implausible value carrier in this particular scenario.

Type:2 Non-minimum value of positive oriented measures or non-maximum value of negative oriented measures for the worst classification. The worst classification occurs when none of the instances is correctly classified. One such scenario is (0, 1, 0, 0), wherein Type-I error is 0.5, which is non-maximum value. Thus, Type-I error will be declared as implausible value carrier in this scenario.

Type:3 Non-maximum value of positive oriented measures or non-minimum value of negative oriented measures for the best classification. The best classification is achieved when all the instances are correctly classified. One such scenario is, (1, 0, 1, 0).

Second and third types are in a mutually exclusive relationship, while they both can have co-occurrence with the first type. This evaluation parameter is named as "plausibility", because implausible value carrier measure either provides insufficient information or does not provide any information about the goodness of a classifier. Plausibility can have effect on the evaluation parameters discussed in the earlier studies, however, exclusive consideration of this aspect is deprived in the literature.

B. CONSISTENCY AND DISCRIMINANCY

Consistency and discriminancy between performance measures are proposed in [109], wherein AUC and Accuracy are compared. However, these two measures can be used to evaluate other performance measures also. Thus, Rosset [14] computes discriminancy between AUC and Error rate, and Haung and Ling [17] use discriminancy and consistency both to evaluate AUC and Accuracy.

Consistency computes the consensus of two performance measures on evaluating different classifiers. It is a symmetric relationship. Two performance measures, say f and g are consistent with each other when comparing two algorithms a and b, if both f and g stipulate that a is better than b. However, if f stipulates that a is better than b and g contradicts that, then f and g are said to be inconsistent.

[4]This notation is used in this paper, where first, second, third and fourth number shows TP, FN, TN, and FP cases respectively.

Discriminancy is an ability of one performance measure over another to discriminate different classifiers. For instance, if f declares a as different (better or worse) than b, while g declares both a and b as equivalent. Then f is more discriminating that g.

Both of these parameters are formally defined in [109] as:

Definition 1 (Consistency): For two measures f, g on domain Ψ, f and g are (strictly) consistent if there exists no a, b such that $f(a) > f(b)$ and $g(a) < g(b)$.

Definition 2 (Discriminancy): For two measures f, g on domain Ψ, f is (strictly) more discriminating than g if there exists a, b belong to Ψ such that $f(a) \neq f(b)$ and $g(a) = g(b)$ and there exists no a, b belong to Ψ such that $g(a) \neq g(b)$ and $f(a) = f(b)$.

Generally, neither *strict* discriminancy nor *strict* consistency exists between performance measures [109]. Therefore, *statistical* consistency and *statistical* discriminancy is subject to be computed, which are defined as:

Definition 3 (Degree of consistency): For two measures f and g on domain Ψ, let

$R = \{(a, b)|a, b \in \Psi\ , f(a) > f(b), g(a) > g(b)\}$ and

$S = \{(a, b)|a, b \in \Psi\ , f(a) > f(b), g(a) < g(b)\}$

$$Degree\ of\ consistency(f, g) = \frac{|R|}{|R| + |S|} \tag{15}$$

We do a minor augmentation to the definition 3 to make it generic enough to cover all possibilities.

Definition 4 (Degree of consistency (augmented)): For two measures f and g on domain Ψ, let

$R = \{(a, b)|a, b \in \Psi\ , f(a) > f(b), g(a) > g(b) \vee f(a) = f(b), g(a) = g(b)\}$

$S = \{(a, b)|a, b \in \Psi\ , f(a) > f(b), g(a) \not> g(b)\} \vee f(a) = f(b), g(a) \neq g(b)$

Definition 5 (Degree of discriminancy): For two measures f and g on domain Ψ, let

$P = \{(a, b)|a, b \in \Psi\ , f(a) > f(b), g(a) = g(b)\}$ and

$Q = \{(a, b)|a, b \in \Psi\ , g(a) > g(b), f(a) = f(b)\}$

$$Degree\ of\ discriminancy(f/g) = \frac{|P|}{|Q|} \tag{16}$$

While computing degree of discriminancy, the ∞ implies strict discriminancy of one measure over another.

VI. EVALUATION OF PERFORMANCE MEASURES

A. PLAUSIBILITY ANALYSIS

1) EXPERIMENTAL DESIGN

The occurrence of zero in any of the four cases potentially leads to implausible values. Thus, all possible scenarios having zero and a non-zero value in the four cases are considered. That makes a total of 14 scenarios, wherein, three scenarios show the best performance and three scenarios show the worst performance. All of these scenarios are shown in Table 3. The performance measures are evaluated against these 14 scenarios.

TABLE 3. The result of plausibility analysis along with the type of implausible value. Top four rows show the values of the four cases (TP, FN TN, and FP), which could be zero or greater than zero; Empty cell shows the plausible value.

TP	>0	>0	0	0	0	0	0	0	0	>0	>0	>0	>0	>0
FN	0	0	0	0	>0	>0	0	>0	>0	0	0	>0	>0	>0
TN	0	>0	>0	0	0	0	>0	>0	>0	0	>0	0	0	>0
FP	0	0	0	>0	>0	0	>0	0	>0	>0	>0	0	>0	0
Precision			Type 1, 3			Type 1, 2		Type 1						
TPR			Type 1, 3	Type 1, 2			Type 1							
TNR	Type 1, 3					Type 1, 2						Type 1		
Accuracy														
F-measure			Type 1, 3	Type 1, 2	Type 1, 2	Type 1, 2	Type 1	Type 1	Type 1					
G-mean1			Type 1, 3	Type 1, 2		Type 1, 2	Type 1	Type 1						
G-mean2	Type 1, 3		Type 1, 3	Type 1, 2		Type 1, 2	Type 1					Type 1		
J coefficient	Type 1, 3		Type 1, 3	Type 1, 2		Type 1, 2	Type 1					Type 1		
FNR			Type 1, 3	Type 1, 2			Type 1							
FPR	Type 1, 3					Type 1, 2						Type 1		
Error rate														
Type-I error					Type 2									
Type-II error					Type 2									
Balance	Type 1, 3		Type 1, 3	Type 1, 2		Type 1, 2	Type 1					Type 1		

2) RESULT AND DISCUSSION

Plausibility analysis concludes that:

1) F-measure is the worst of all, by having implausible values in seven scenarios, including implausible value of Type-1 and Type-3 in one best classification and Type-1 and Type-2 in all three worst classifications. It is further illustrated that F-measure cannot identify the worst classification.
2) F-measure is followed by G-mean2, J coefficient and Balance by having implausible values in six scenarios.
3) Accuracy and Error rate are the safest of all by having never implausible value.
4) Complement measures have the same behavior as that of their countermeasures. Like, Accuracy and Error rate both, never produce implausible value. Likewise, TPR and FNR produce implausible value three times.
5) Derived measures always inherit the Type-3 implausible value from their base measures with some time addition of their own. Like, F-measure produces Type-3 implausible value seven times, where five are taken from its base measures.

The implication of the above findings is that the measures having more occurrences of implausible value should not be used where there is a high chance of occurrence of zero in any of the four cases.

B. CONSISTENCY AND DISCRIMINANCY ANALYSES

1) EXPERIMENTAL DESIGN

Consistency and discriminancy analyses are conducted between two classifiers. Logically, diversity between results makes more rigorous analyses. In order to achieve this, we performed following steps to get diverse results from hypothetical classifiers.

Step:1 We take two types of datasets; balanced and imbalanced, as is done in [17]. In fact, this leads to three datasets; In the first dataset, $P = N$ (Where, P and N show total number of positive and negative instances, respectively), in the second dataset $P < N$ and third dataset has $P > N$.

a) Dataset-1: $P = N, n = 12$; $P = 6, N = 6$
b) Dataset-2: $P < N, n = 30$; $P = 12, N = 18$
c) Dataset-3: $P > N, n = 30$; $P = 18, N = 12$

The reason for specifying the value of n, P and N is described in Step-3.

Step:2 In this step, we generate diverse conditions that a result is supposed to satisfy. Diverse conditions imply a different relationship between pairs of cases. Pairs of cases are (TP, FN), (TP, TN), (TP, FP), (FN, TN), (FN, FP), and (TN, FP). Each pair can have three possible relationships between them $=$, $<$, and $>$. Each of the six pairs are combined to make a single condition like, $\{TP > FP \wedge TP = FN \wedge TP < TN \wedge TN > FP \wedge TN > FN \wedge FP < FN\}$ is one condition for single result to satisfy. Eventually, the pairs with all possible relationships make a total of 729 unique conditions. Of course some conditions cannot be satisfied which are identified and dropped in the next step.

Step:3 In this step, we assign values to four cases a way that maximum conditions can be satisfied. To assign values to four cases, we develop a tool that uses all values ranges from one [5] to 20. Using the tool, conditions which could not be satisfied are automatically identified and dropped. Eventually, in the first dataset, where $P = N$, 17 conditions are satisfied. In the second and third dataset where $P < N$ and $P > N$ respectively, 41 conditions could be satisfied. All satisfied conditions in each dataset and their values in corresponding cases are shown in Table 4. In the first dataset, n is taken as 12 because it is the smallest number that could satisfy as many conditions as when n is taken as 80. In the second and third dataset, n is taken as 30 for the same reason. Results against each dataset are assumed to be results from hypothetical classifiers. Intuitively speaking, the result of 17 classifiers are taken from the first dataset and

[5] We started with one to avoid an occurrence of zero value in any case for it could cause implausible value.

TABLE 4. The result from 99 classifiers with corresponding satisfied conditions; The topmost block is the result from the balanced dataset, whereas middle and lower blocks show results when $P < N$ and $P > N$ respectively.

	TP	FN	TN	FP	Satisfied Conditions
Results of 17 classifiers on balanced dataset (P = N = 6)	3	3	3	3	TP = FP∧TP = FN∧TP = TN∧TN = FP∧TN = FN∧FP = FN
	1	5	1	5	TP<FP∧TP<FN∧TP = TN∧TN<FP∧TN<FN∧FP = FN
	1	5	5	1	TP = FP∧TP<FN∧TP<TN∧TN>FP∧TN = FN∧FP<FN
	5	1	1	5	TP = FP∧TP>FN∧TP>TN∧TN<FP∧TN = FN∧FP>FN
	5	1	5	1	TP>FP∧TP>FN∧TP = TN∧TN>FP∧TN>FN∧FP = FN
	1	5	3	3	TP<FP∧TP<FN∧TP<TN∧TN = FP∧TN<FN∧FP<FN
	3	3	1	5	TP<FP∧TP = FN∧TP>TN∧TN<FP∧TN<FN∧FP>FN
	3	3	5	1	TP>FP∧TP = FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	5	1	3	3	TP>FP∧TP>FN∧TP>TN∧TN = FP∧TN>FN∧FP>FN
	1	5	2	4	TP<FP∧TP<FN∧TP<TN∧TN<FP∧TN<FN∧FP<FN
	1	5	4	2	TP<FP∧TP<FN∧TP<TN∧TN>FP∧TN<FN∧FP<FN
	2	4	1	5	TP<FP∧TP<FN∧TP>TN∧TN<FP∧TN<FN∧FP>FN
	2	4	5	1	TP>FP∧TP<FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	4	2	1	5	TP<FP∧TP>FN∧TP>TN∧TN<FP∧TN<FN∧FP>FN
	4	2	5	1	TP>FP∧TP>FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	5	1	2	4	TP>FP∧TP>FN∧TP>TN∧TN<FP∧TN>FN∧FP>FN
	5	1	4	2	TP>FP∧TP>FN∧TP>TN∧TN>FP∧TN>FN∧FP>FN
Results of 41 classifiers on imbalanced data when P<N (P = 12 and N = 18)	3	9	9	9	TP<FP∧TP<FN∧TP<TN∧TN = FP∧TN = FN∧FP = FN
	9	3	9	9	TP = FP∧TP>FN∧TP = TN∧TN = FP∧TN>FN∧FP>FN
	6	6	9	9	TP<FP∧TP = FN∧TP<TN∧TN = FP∧TN>FN∧FP>FN
	6	6	6	12	TP<FP∧TP = FN∧TP = TN∧TN<FP∧TN = FN∧FP>FN
	6	6	12	6	TP = FP∧TP = FN∧TP<TN∧TN>FP∧TN>FN∧FP = FN
	6	6	1	17	TP<FP∧TP = FN∧TP>TN∧TN<FP∧TN<FN∧FP>FN
	6	6	17	1	TP>FP∧TP = FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	1	11	7	11	TP<FP∧TP<FN∧TP<TN∧TN<FP∧TN<FN∧FP = FN
	1	11	11	7	TP<FP∧TP<FN∧TP<TN∧TN>FP∧TN = FN∧FP<FN
	11	1	7	11	TP = FP∧TP>FN∧TP>TN∧TN<FP∧TN>FN∧FP>FN
	11	1	11	7	TP>FP∧TP>FN∧TP = TN∧TN>FP∧TN>FN∧FP>FN
	1	11	9	9	TP<FP∧TP<FN∧TP<TN∧TN = FP∧TN<FN∧FP<FN
	11	1	9	9	TP>FP∧TP>FN∧TP>TN∧TN = FP∧TN>FN∧FP>FN
	1	11	1	17	TP<FP∧TP<FN∧TP = TN∧TN<FP∧TN<FN∧FP>FN
	1	11	17	1	TP = FP∧TP<FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	11	1	1	17	TP<FP∧TP>FN∧TP>TN∧TN<FP∧TN = FN∧FP>FN
	11	1	17	1	TP>FP∧TP>FN∧TP<TN∧TN>FP∧TN>FN∧FP = FN
	2	10	1	17	TP<FP∧TP<FN∧TP>TN∧TN<FP∧TN<FN∧FP>FN
	2	10	17	1	TP>FP∧TP<FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	10	2	1	17	TP<FP∧TP>FN∧TP>TN∧TN<FP∧TN<FN∧FP>FN
	10	2	17	1	TP>FP∧TP>FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	1	11	2	16	TP<FP∧TP<FN∧TP<TN∧TN<FP∧TN<FN∧FP>FN
	1	11	16	2	TP<FP∧TP<FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	11	1	2	16	TP<FP∧TP>FN∧TP>TN∧TN<FP∧TN>FN∧FP>FN
	11	1	16	2	TP>FP∧TP>FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	4	8	9	9	TP<FP∧TP<FN∧TP<TN∧TN = FP∧TN>FN∧FP>FN
	8	4	9	9	TP<FP∧TP>FN∧TP<TN∧TN = FP∧TN>FN∧FP>FN
	4	8	8	10	TP<FP∧TP<FN∧TP<TN∧TN<FP∧TN = FN∧FP>FN
	4	8	10	8	TP<FP∧TP<FN∧TP<TN∧TN>FP∧TN>FN∧FP = FN
	8	4	8	10	TP<FP∧TP>FN∧TP = TN∧TN<FP∧TN>FN∧FP>FN
	8	4	10	8	TP = FP∧TP>FN∧TP<TN∧TN>FP∧TN>FN∧FP>FN
	6	6	7	11	TP<FP∧TP = FN∧TP<TN∧TN<FP∧TN>FN∧FP>FN
	6	6	11	7	TP<FP∧TP = FN∧TP<TN∧TN>FP∧TN>FN∧FP>FN
	1	11	8	10	TP<FP∧TP<FN∧TP<TN∧TN<FP∧TN<FN∧FP<FN
	1	11	10	8	TP<FP∧TP<FN∧TP<TN∧TN>FP∧TN<FN∧FP<FN
	11	1	8	10	TP>FP∧TP>FN∧TP>TN∧TN<FP∧TN>FN∧FP>FN
	11	1	10	8	TP>FP∧TP>FN∧TP>TN∧TN>FP∧TN>FN∧FP>FN
	5	7	8	10	TP<FP∧TP<FN∧TP<TN∧TN<FP∧TN>FN∧FP>FN
	5	7	10	8	TP<FP∧TP<FN∧TP<TN∧TN>FP∧TN>FN∧FP>FN
	7	5	8	10	TP<FP∧TP>FN∧TP<TN∧TN<FP∧TN>FN∧FP>FN
	7	5	10	8	TP<FP∧TP>FN∧TP<TN∧TN>FP∧TN>FN∧FP>FN
Results of 41 classifiers on imbalanced data when P>N (P = 18 and N = 12)	9	9	3	9	TP = FP∧TP = FN∧TP>TN∧TN<FP∧TN<FN∧FP = FN
	9	9	9	3	TP>FP∧TP = FN∧TP = TN∧TN>FP∧TN = FN∧FP<FN
	9	9	6	6	TP>FP∧TP = FN∧TP>TN∧TN = FP∧TN<FN∧FP<FN
	6	12	6	6	TP = FP∧TP<FN∧TP = TN∧TN = FP∧TN<FN∧FP<FN
	12	6	6	6	TP>FP∧TP>FN∧TP>TN∧TN = FP∧TN = FN∧FP = FN
	1	17	6	6	TP<FP∧TP<FN∧TP<TN∧TN = FP∧TN<FN∧FP<FN
	17	1	6	6	TP>FP∧TP>FN∧TP>TN∧TN = FP∧TN>FN∧FP>FN
	7	11	1	11	TP<FP∧TP<FN∧TP>TN∧TN<FP∧TN<FN∧FP = FN
	7	11	11	1	TP>FP∧TP<FN∧TP<TN∧TN>FP∧TN = FN∧FP<FN
	11	7	1	11	TP = FP∧TP>FN∧TP>TN∧TN<FP∧TN<FN∧FP>FN
	11	7	11	1	TP>FP∧TP>FN∧TP = TN∧TN>FP∧TN>FN∧FP>FN
	9	9	1	11	TP<FP∧TP = FN∧TP>TN∧TN<FP∧TN<FN∧FP>FN
	9	9	11	1	TP>FP∧TP = FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	1	17	1	11	TP<FP∧TP<FN∧TP = TN∧TN<FP∧TN<FN∧FP<FN
	1	17	11	1	TP = FP∧TP<FN∧TP<TN∧TN>FP∧TN<FN∧FP<FN
	17	1	1	11	TP>FP∧TP>FN∧TP>TN∧TN<FP∧TN = FN∧FP>FN
	17	1	11	1	TP>FP∧TP>FN∧TP>TN∧TN>FP∧TN>FN∧FP = FN
	1	17	2	10	TP<FP∧TP<FN∧TP<TN∧TN<FP∧TN<FN∧FP<FN
	1	17	10	2	TP<FP∧TP<FN∧TP<TN∧TN>FP∧TN<FN∧FP<FN
	17	1	2	10	TP>FP∧TP>FN∧TP>TN∧TN<FP∧TN>FN∧FP>FN
	17	1	10	2	TP>FP∧TP>FN∧TP>TN∧TN>FP∧TN>FN∧FP>FN
	2	16	1	11	TP<FP∧TP<FN∧TP>TN∧TN<FP∧TN<FN∧FP<FN
	2	16	11	1	TP>FP∧TP<FN∧TP<TN∧TN>FP∧TN<FN∧FP<FN
	16	2	1	11	TP>FP∧TP>FN∧TP>TN∧TN<FP∧TN>FN∧FP<FN
	16	2	11	1	TP>FP∧TP>FN∧TP>TN∧TN>FP∧TN>FN∧FP<FN
	9	9	4	8	TP>FP∧TP = FN∧TP>TN∧TN<FP∧TN<FN∧FP<FN
	9	9	8	4	TP>FP∧TP = FN∧TP>TN∧TN>FP∧TN<FN∧FP<FN
	8	10	4	8	TP = FP∧TP<FN∧TP>TN∧TN<FP∧TN<FN∧FP<FN
	8	10	8	4	TP>FP∧TP<FN∧TP = TN∧TN>FP∧TN<FN∧FP<FN
	10	8	4	8	TP>FP∧TP>FN∧TP>TN∧TN<FP∧TN<FN∧FP = FN
	10	8	8	4	TP>FP∧TP>FN∧TP>TN∧TN>FP∧TN = FN∧FP<FN
	7	11	6	6	TP>FP∧TP<FN∧TP>TN∧TN = FP∧TN<FN∧FP<FN
	11	7	6	6	TP>FP∧TP>FN∧TP>TN∧TN = FP∧TN<FN∧FP<FN
	8	10	1	11	TP<FP∧TP<FN∧TP>TN∧TN<FP∧TN<FN∧FP>FN
	8	10	11	1	TP>FP∧TP<FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	10	8	1	11	TP<FP∧TP>FN∧TP>TN∧TN<FP∧TN<FN∧FP>FN
	10	8	11	1	TP>FP∧TP>FN∧TP<TN∧TN>FP∧TN>FN∧FP<FN
	8	10	5	7	TP>FP∧TP<FN∧TP>TN∧TN<FP∧TN<FN∧FP<FN
	8	10	7	5	TP>FP∧TP<FN∧TP>TN∧TN>FP∧TN<FN∧FP<FN
	10	8	5	7	TP>FP∧TP>FN∧TP>TN∧TN<FP∧TN<FN∧FP<FN
	10	8	7	5	TP>FP∧TP>FN∧TP>TN∧TN>FP∧TN<FN∧FP<FN

results from 41 classifiers are taken from second and third dataset each. Table 4 can be viewed as results from 99 different hypothetical classifiers.

Step:4 In this step, all the performance measures are computed against results from 99 classifiers.

Step:5 Finally, we make all possible pairs of results (from every classifier), within a dataset. That makes a total of 136 pairs in the balanced dataset and 820 pairs in each imbalanced datasets. That would allow applying consistency and discriminancy analyses on each pair.

2) RESULTS AND DISCUSSION ON CONSISTENCY ANALYSIS

The result of consistency on balanced and imbalanced datasets are shown in Table 5. This experiment concludes that;

1) Generally, the degree of consistency is influenced by the relationship between P and N.
2) Complementing measures always have 100% consistency between them.
3) Other than complementing measures, the following pairs are strictly consistent across the datasets;
   a) {TPR, Type-II error}
   b) {TNR, Type-I error}
   c) {FNR, Type-II error}
   d) {FPR, Type-I error}
4) There are four pairs of performance measures having strict consistency between them only in the balanced dataset.
   a) {F-measure, G-mean1}
   b) {G-mean2, Balance}
   c) {Accuracy, J coefficient}
   d) {J coefficient, Error rate}
5) Strict consistency holds transitive relationship in the respective dataset.
6) All the strictly consistent performance measures in the imbalanced datasets are found strictly consistent in balanced dataset also.

Above findings imply that the strictly consistent performance measures are not supposed to be used together to prevent redundant evaluation of machine learning models, as is done in different studies [48], [83] etc. Instead, least consistent performance measures may be used. In addition to that, P to N ratio in the dataset should be considered while choosing the performance measures to compare different models.

3) RESULTS AND DISCUSSION ON DISCRIMINANCY ANALYSIS

Discriminancy analysis is done in three different dimensions. Firstly, we compute the degree of discriminancy between all possible pairs of the performance measures (see Table 6). Secondly, we compute discriminancy ability of each performance measure irrespective of other. Whose results are drawn in Figure 2. Finally, we try to generalize the scenarios when

TABLE 5. Degree of consistency between performance measures. The topmost block is the result from the balanced dataset, whereas middle and lower blocks show results when $P < N$ and $P > N$ respectively.

	Balance	Error rate	Type-II error	Type-I error	FPR	FNR	*J* coefficient	G-mean2	G-mean1	F-measure	Accuracy	TNR	TPR
Precision	0.91	0.93	0.65	0.65	0.65	0.65	0.93	0.91	0.87	0.87	0.93	0.65	0.65
TPR	0.63	0.63	1	0.32	0.32	1	0.63	0.63	0.78	0.78	0.63	0.32	
TNR	0.63	0.63	0.32	1	1	0.32	0.63	0.63	0.54	0.54	0.63		
Accuracy	0.99	1	0.63	0.63	0.63	0.63	1	0.99	0.85	0.85			
F-measure	0.85	0.85	0.78	0.54	0.54	0.78	0.85	0.85	1				
G-mean1	0.85	0.85	0.78	0.54	0.54	0.78	0.85	0.85					
G-mean2	1	0.99	0.63	0.63	0.63	0.63	0.99						
J coefficient	0.99	1	0.63	0.63	0.63	0.63							
FNR	0.63	0.63	1	0.32	0.32								
FPR	0.63	0.63	0.32	1									
Type-I error	0.63	0.63	0.32										
Type-II error	0.63	0.63											
Error rate	0.99												
Precision	0.88	0.91	0.71	0.66	0.66	0.71	0.93	0.88	0.87	0.88	0.91	0.66	0.71
TPR	0.73	0.64	1	0.39	0.39	1	0.73	0.73	0.84	0.83	0.64	0.39	
TNR	0.66	0.7	0.39	1	1	0.39	0.64	0.65	0.55	0.55	0.7		
Accuracy	0.88	1	0.64	0.7	0.7	0.64	0.9	0.89	0.8	0.81			
F-measure	0.9	0.81	0.83	0.55	0.55	0.83	0.89	0.89	0.99				
G-mean1	0.89	0.8	0.84	0.55	0.55	0.84	0.89	0.89					
G-mean2	0.98	0.89	0.73	0.65	0.65	0.73	0.92						
J coefficient	0.92	0.9	0.73	0.64	0.64	0.73							
FNR	0.73	0.64	1	0.39	0.39								
FPR	0.66	0.7	0.39	1									
Type-I error	0.66	0.7	0.39										
Type-II error	0.73	0.64											
Error rate	0.88												
Precision	0.9	0.91	0.66	0.71	0.71	0.66	0.93	0.91	0.84	0.82	0.91	0.71	0.66
TPR	0.66	0.7	1	0.39	0.39	1	0.64	0.65	0.82	0.83	0.7	0.39	
TNR	0.73	0.64	0.39	1	1	0.39	0.73	0.73	0.57	0.56	0.64		
Accuracy	0.88	1	0.7	0.64	0.64	0.7	0.9	0.89	0.88	0.87			
F-measure	0.82	0.87	0.83	0.56	0.56	0.83	0.81	0.82	0.99				
G-mean1	0.83	0.88	0.82	0.57	0.57	0.82	0.82	0.83					
G-mean2	0.98	0.89	0.65	0.73	0.73	0.65	0.92						
J coefficient	0.92	0.9	0.64	0.73	0.73	0.64							
FNR	0.66	0.7	1	0.39	0.39								
FPR	0.73	0.64	0.39	1									
Type-I error	0.73	0.64	0.39										
Type-II error	0.66	0.7											
Error rate	0.88												

the respective performance measure alone fails to discriminate two classifiers (see Table 7). The discriminancy analysis concludes that:

1) F-measure is strictly more discriminant than rest of the performance measures excluding G-mean1, across all datasets. Likewise, G-mean1 is strictly more discriminant than the rest of the performance measures excluding F-measures in all the datasets.
2) In the balanced dataset, strict discriminancy of Precision, G-mean2 and, Balance are found over Accuracy, *J* coefficient, and Error rate.
3) In the imbalanced datasets when $P > N$, Balance has strict discriminancy over G-mean2 and *J* coefficient.
4) Complementing measures have zero degree of discriminancy over each other. Moreover, they share same discriminancy relationship with other measures as that of their countermeasures.
5) Derived measures have a higher degree of discriminancy than their base measures.

Our next focus is to individually evaluate the performance measures to discriminate 99 classifiers. Eventually, we conclude results shown in Figure 2. Figure depicts that none of the performance measures successfully discriminate all the classifiers. Moreover, the best out of the performance measures are F-measure and G-mean1, which discriminate 98 classifiers, yet not effective enough to discriminate all the classifiers. This small experiment spurs to figure out the cases

TABLE 6. Degree of discriminancy between performance measures in the first row over the performance measure in the first column. The topmost block is the result from the balanced dataset, whereas middle and lower blocks show results when $P < N$ and $P > N$ respectively.

	Precision	TPR	TNR	Accuracy	F-measure	G-mean1	G-mean2	*J* coefficient	FNR	FPR	Type-I error	Type-II error	Error rate	Balance
Precision		0.12	0.12	0	∞	∞	0.33	0	0.12	0.12	0.12	0.12	0	0.33
TPR	8.33		1	2.78	∞	∞	3.57	2.78	0	1	1	0	2.78	3.57
TNR	8.33	1		2.78	∞	∞	3.57	2.78	1	0	0	1	2.78	3.57
Accuracy	∞	0.36	0.36		∞	∞	∞	0	0.36	0.36	0.36	0.36	0	∞
F-measure	0	0	0	0		0	0	0	0	0	0	0	0	0
G-mean1	0	0	0	0	0		0	0	0	0	0	0	0	0
G-mean2	3	0.28	0.28	0	∞	∞		0	0.28	0.28	0.28	0.28	0	0
J coefficient	∞	0.36	0.36	0	∞	∞	∞		0.36	0.36	0.36	0.36	0	∞
FNR	8.33	0	1	2.78	∞	∞	3.57	2.78		1	1	0	2.78	3.57
FPR	8.33	1	0	2.78	∞	∞	3.57	2.78	1		0	1	2.78	3.57
Type-I error	8.33	1	0	2.78	∞	∞	3.57	2.78	1	0		1	2.78	3.57
Type-II error	8.33	0	1	2.78	∞	∞	3.57	2.78	0	1	1		2.78	3.57
Error rate	∞	0.36	0.36	0	∞	∞	∞	0	0.36	0.36	0.36	0.36		∞
Balance	3	0.28	0.28	0	∞	∞	0	0	0.28	0.28	0.28	0.28	0	
Precision		0.16	0.2	0.21	16	16	3	1.14	0.16	0.2	0.2	0.16	0.21	8
TPR	6.44		1.3	2.64	103	103	17.17	7.36	0	1.3	1.3	0	2.64	51.5
TNR	4.94	0.77		2.03	79	79	13.17	5.64	0.77	0	0	0.77	2.03	39.5
Accuracy	4.83	0.38	0.49		39	39	12	2.79	0.38	0.49	0.49	0.38	0	19.5
F-measure	0.06	0.01	0.01	0.03		1	0.17	0.07	0.01	0.01	0.01	0.01	0.03	0.5
G-mean1	0.06	0.01	0.01	0.03	1		0.17	0.07	0.01	0.01	0.01	0.01	0.03	0.5
G-mean2	0.33	0.06	0.08	0.08	6	6		0.33	0.06	0.08	0.08	0.06	0.08	∞
J coefficient	0.88	0.14	0.18	0.36	14	14	3		0.14	0.18	0.18	0.14	0.36	∞
FNR	6.44	0	1.3	2.64	103	103	17.17	7.36		1.3	1.3	0	2.64	51.5
FPR	4.94	0.77	0	2.03	79	79	13.17	5.64	0.77		0	0.77	2.03	39.5
Type-I error	4.94	0.77	0	2.03	79	79	13.17	5.64	0.77	0		0.77	2.03	39.5
Type-II error	6.44	0	1.3	2.64	103	103	17.17	7.36	0	1.3	1.3		2.64	51.5
Error rate	4.83	0.38	0.49	0	39	39	12	2.79	0.38	0.49	0.49	0.38		19.5
Balance	0.13	0.02	0.03	0.05	2	2	0	0	0.02	0.03	0.03	0.02	0.05	
Precision		0.2	0.16	0.21	∞	∞	3	1.14	0.2	0.16	0.16	0.2	0.21	8
TPR	4.94		0.77	2.03	∞	∞	13.17	5.64	0	0.77	0.77	0	2.03	39.5
TNR	6.44	1.3		2.64	∞	∞	17.17	7.36	1.3	0	0	1.3	2.64	51.5
Accuracy	4.83	0.49	0.38		∞	∞	12	2.79	0.49	0.38	0.38	0.49	0	19.5
F-measure	0	0	0	0		0	0	0	0	0	0	0	0	0
G-mean1	0	0	0	0	0		0	0	0	0	0	0	0	0
G-mean2	0.33	0.08	0.06	0.08	∞	∞		0.33	0.08	0.06	0.06	0.08	0.08	∞
J coefficient	0.88	0.18	0.14	0.36	∞	∞	3		0.18	0.14	0.14	0.18	0.36	∞
FNR	4.94	0	0.77	2.03	∞	∞	13.17	5.64		0.77	0.77	0	2.03	39.5
FPR	6.44	1.3	0	2.64	∞	∞	17.17	7.36	1.3		0	1.3	2.64	51.5
Type-I error	6.44	1.3	0	2.64	∞	∞	17.17	7.36	1.3	0		1.3	2.64	51.5
Type-II error	4.94	0	0.77	2.03	∞	∞	13.17	5.64	0	0.77	0.77		2.03	39.5
Error rate	4.83	0.49	0.38	0	∞	∞	12	2.79	0.49	0.38	0.38	0.49		19.5
Balance	0.13	0.03	0.02	0.05	∞	∞	0	0	0.03	0.02	0.02	0.03	0.05	

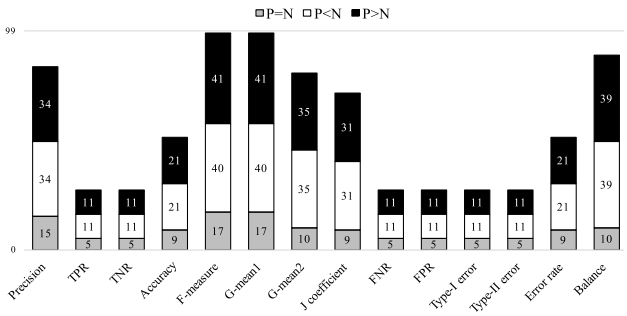


FIGURE 2. Number of discriminated classifiers by the performance measures.

where the performance measures fail to discriminate two different classifiers. We conclude that performance measures have an inherent property not to discriminate classifiers having certain common properties. These properties are written in Table 7. Middle column of the table shows conditions, if satisfied by a classifier *a* and *b*, then the respective measure would not be able to discriminate *a* and *b*.

4) COMBINED ANALYSIS OF CONSISTENCY AND DISCRIMINANCY

Ling *et al.* design two rules for combined effect of consistency and discriminancy between performance measures [109].

TABLE 7. Non-discriminable scenarios for performance measures.

Performance measure	Non-discriminating scenarios in the results from classifier a and b	Example of non-discriminable results
Precision	$a(TP) : a(FP) = b(TP) : b(FP)$	$a(6,6,6,12), b(1,11,16,2)$
TPR	$a(TP) = b(TP)$ or $a(FN) = b(FN)$	$a(1,5,1,5), b(1,5,3,3)$
TNR	$a(TN) = b(TN)$ or $a(FP) = b(FP)$	$a(1,5,5,1), b(2,4,5,1)$
Accuracy	$sum(a(TP), a(TN)) = sum(b(TP), b(TN))$ or $sum(a(FN), a(FP)) = sum(b(FN), b(FP))$	$a(4,2,5,1), b(5,1,4,2)$
F-measure	$2 \times a(TP) : sum(a(TP), a(FP)) = 2 \times b(TP) : sum(b(TP), b(FP))$	$a(6,6,6,12), b(5,7,10,8)$
G-mean1	$(a(TP))^2 : sum(a(TP), a(FP)) = (b(TP))^2 : sum(b(TP), b(FP))$	$a(2,10,17,1), b(4,8,10,8)$
G-mean2	$a(TP) : b(TN) = a(TN) : b(TP)$ or $a(FN) : b(FP) = a(FP) : b(FN)$	$a(6,6,12,6), b(8,4,9,9)$
J coefficient	$\begin{cases} a(TP) = b(TN) \text{ and } a(TN) = b(TP) & if(P = N) \\ a(TP) : P = b(TN) : N \text{ and } a(TN) : N = b(TP) : P & otherwise \end{cases}$	$a(1,5,5,1), b(5,1,1,5)$ $a(6,6,11,7), b(8,4,8,10)$
FNR	$a(TP) = b(TP)$ or $a(FN) = b(FN)$	$a(5,1,5,1), b(5,1,4,2)$
FPR	$a(TN) = b(TN)$ or $a(FP) = b(FP)$	$a(5,1,5,1), b(4,2,5,1)$
Type-I error	$a(TN) = b(TN)$ or $a(FP) = b(FP)$	$a(5,1,5,1), b(4,2,5,1)$
Type-II error	$a(TP) = b(TP)$ or $a(FN) = b(FN)$	$a(5,1,5,1), b(5,1,4,2)$
Error rate	$sum(a(TP), a(TN)) = sum(b(TP), b(TN))$ or $sum(a(TP), af(TN)) = sum(b(TP), b(TN))$	$a(5,1,4,2), b(4,2,5,1)$
Balance	$\begin{cases} a(TN) = b(TP) \text{ and } a(TP) = b(TN) & if(P = N) \\ a(TN) : N = b(TP) : P \text{ and } a(TP) : N = b(TN) : P & otherwise \end{cases}$	$a(1,5,5,1), b(5,1,1,5)$ $a(6,12,6,6), b(9,9,4,8)$

Rule-1: If $C_{(f,g)} = 1.0$ and $D_{(f/g)} = \infty$, then f is strictly better than g.
Rule-2: If $C_{(f,g)} > 0.5$ and $D_{(f/g)} > 1$, then f is statistically better than g.
Combined analysis of consistency and discriminancy infers the overall goodness of one performance measure over another. Table 8 depicts the comparison between performance measures in the first column and performance measures on the header row. The former is better than the later one if the specified cell contains "Yes". The table illustrates that the following order from better to worst prevails in the balanced dataset:
1^{st}: F-measures/G-mean1
2^{nd}: Precision
3^{rd}: G-mean2
4^{th}: Balance
5^{th}: J coefficient
6^{th}: Accuracy / Error rate
7^{th}: TPR / FNR / TNR / FPR / Type-I error / Type-II error
The above order is slightly modified in the imbalanced datasets.
1^{st}: F-measures/G-mean1
2^{nd}: G-mean2
3^{rd}: Balance
4^{th}: J coefficient
5^{th}: Precision
6^{th}: Accuracy/Error rate
7^{th}: TPR / FNR / TNR / FPR / Type-I error / Type-II error

The base measures are identified as the worst of all the measures with the exception of Precision, which is quite unstable, as it degrades from 2^{nd} to 5^{th} position as we move from balanced to the imbalanced datasets. F-measure and G-mean1 are the best of all measures. However, if there is a chance of occurrence of zero in any of the case, then next better measures may be used.

TABLE 8. Combined result of consistency and discriminancy.

	Precision	TPR / TNR / FPR / FNR	Accuracy / Error rate	G-mean2	J coefficient	Type-I error / Type-II error	Balance	Dataset
Accuracy/Error rate								
F-measure/G-mean1	Yes		Yes	Yes	Yes		Yes	
G-mean2			Yes		Yes			
Balance			Yes		Yes			
Precision			Yes	Yes	Yes		Yes	*Balanced*
J coefficient								
Accuracy/Error rate								
F-measure/G-mean1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
G-mean2	Yes				Yes			
Balance	Yes			Yes	Yes			
Precision			Yes					*Imbalanced*
J coefficient	Yes							

VII. COMPARISON WITH OTHER STUDIES

Plausibility analysis deduces the outperformance of Accuracy and Error rate as compared to rest of the performance measures. Though AUC has been compared with Accuracy by many studies [10], [11], [15], [17] and with Error rate by some others [13], [14], however, evaluation in plausibility and in comparison with the performance measure we take, have not been done before. Besides this, Accuracy / Error rate

is found to be less discriminating. So, when it comes to the evaluation of single model then Accuracy / Error rate may be used as is done by many authors, whereas, for the comparison of different models, it is not encouraged. Same conclusion is drawn in [14].

Selection of performance measure to compare different classifiers is subjected by many studies [6], [20], [23]. Most of these studies not clearly declare any performance measure or a pair of performance measure (from the list we have drawn) to be used in classifiers' evaluation. Amongst them, [20] declares that Precision and Type-I error are closely related, our work agrees that both of these two measures are found consistent in 1164 comparisons out of 1776.

VIII. CONCLUSION

Our experiments declare that F-measure and G-mean1 are equally the most discriminating performance measures, hence the most suitable to evaluate two different classifiers. Besides this, both of these measures are found to be the least scorer in plausibility analysis. So, it is good to use one of them when there is NO possibility of occurrence of zero in any of the four cases. However, where there is a chance of occurrence of zero in any of the case, Precision would be relatively a good choice for having more plausibility score and more discriminating.

IX. FUTURE GUIDELINES

For future work, we plan to evaluate the performance measures used in regression like Completeness [57], Average absolute error [110] etc. Moreover, our work is limited to evaluate performance measure used in SFP, however, performance measures used in other domains may also be evaluated.

REFERENCES

[1] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process," in *Proc. 7th Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng.*, 2009, pp. 91–100.

[2] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7346–7354, May 2009.

[3] S. Beecham, T. Hall, D. Bowes, D. Gray, S. Counsell, and S. Black, "A systematic review of fault prediction approaches used in software engineering," Lero, Dubrovnik, Croatia, Tech. Rep. Lero-TR-2010-04, 2010.

[4] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Amsterdam, The Netherlands: Elsevier, 2011.

[5] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Appl. Soft Comput.*, vol. 27, pp. 504–518, Feb. 2015.

[6] E. Arisholm, L. C. Briand, and E. B. Johannessen, "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models," *J. Syst. Softw.*, vol. 83, no. 1, pp. 2–17, Jan. 2010.

[7] E. J. Weyuker, T. J. Ostrand, and R. M. Bell, "Comparing the effectiveness of several modeling methods for fault prediction," *Empirical Softw. Eng.*, vol. 15, no. 3, pp. 277–295, Jun. 2010.

[8] C. Catal, "Software fault prediction: A literature review and current trends," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4626–4636, 2011.

[9] D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: A systematic literature review," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1397–1418, Aug. 2013.

[10] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognit.*, vol. 30, no. 7, pp. 1145–1159, Jul. 1997.

[11] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Mach. Learn.*, vol. 30, nos. 2–3, pp. 195–215, Feb. 1998.

[12] P. A. Flach, "The geometry of ROC space: Understanding machine learning metrics through ROC isometrics," in *Proc. 20th Int. Conf. Mach. Learn.* Menlo Park, CA, USA: AAAI Press, 2003, pp. 194–201.

[13] C. Cortes and M. Mohri, "AUC optimization vs. error rate minimization," in *Proc. 16th Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Cambridge, MA, USA: MIT Press, 2003, pp. 313–320.

[14] S. Rosset, "Model selection via the AUC," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 89.

[15] R. Caruana and A. Niculescu-Mizil, "Data mining in metric space: An empirical analysis of supervised learning performance criteria," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 69–78.

[16] J. Fürnkranz and P. A. Flach, "ROC 'n' rule learning–towards a better understanding of covering algorithms," *Mach. Learn.*, vol. 58, no. 1, pp. 39–77, Jan. 2005.

[17] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 3, pp. 299–310, Mar. 2005.

[18] J. Davis and M. Goadrich, "The relationship between Precision–Recall and ROC curves," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*. New York, NY, USA: ACM, 2006, pp. 233–240.

[19] J. Huang and C. X. Ling, "Constructing new and better evaluation measures for machine learning," in *Proc. IJCAI*, 2007, pp. 859–864.

[20] T. J. Ostrand and E. J. Weyuker, "How to measure success of fault prediction models," in *Proc. 4th Int. Workshop Softw. Quality Assurance, Conjunct 6th ESEC/FSE Joint Meeting*, 2007, pp. 25–30.

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

[22] 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, Jan. 2009.

[23] Y. Jiao and P. Du, "Performance measures in evaluating machine learning based bioinformatics predictors for classifications," *Quant. Biol.*, vol. 4, no. 4, pp. 320–330, Dec. 2016.

[24] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to predict severity of software vulnerability using only vulnerability description," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2017, pp. 125–136.

[25] Y. Zhou and H. Leung, "Empirical analysis of object-oriented design metrics for predicting high and low severity faults," *IEEE Trans. Softw. Eng.*, vol. 32, no. 10, pp. 771–789, Oct. 2006.

[26] B. Kolo, *Binary and Multiclass Classification*. Weatherford, OK, USA: Weatherford Press, 2010.

[27] T. Menzies, J. DiStefano, A. Orrego, and R. Chapman, "Assessing predictors of software defects," in *Proc. Workshop Predictive Softw. Models*, 2004, pp. 1–4.

[28] L. Guo, B. Cukic, and H. Singh, "Predicting fault prone modules by the Dempster–Shafer belief networks," in *Proc. 18th IEEE Int. Conf. Automated Softw. Eng.*, Oct. 2003, pp. 249–252.

[29] A. Mahaweerawat, P. Sophatsathit, and C. Lursinsap, "Software fault prediction using fuzzy clustering and radial-basis function network," in *Proc. Int. Conf. Intell. Technol., InTech/VJFuzzy*, Vietnam, Dec. 2002.

[30] D. Grossman and P. Domingos, "Learning Bayesian network classifiers by maximizing conditional likelihood," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 46.

[31] A. Kent, M. M. Berry, F. U. Luehrs, Jr., and J. W. Perry, "Machine literature searching VIII. Operational criteria for designing information retrieval systems," *Amer. Document.*, vol. 6, no. 2, pp. 93–101, 1955.

[32] Y. Singh, A. Kaur, and R. Malhotra, "Prediction of fault-prone software modules using statistical and machine learning methods," *Int. J. Comput. Appl.*, vol. 1, no. 22, pp. 8–15, 2010.

[33] L. C. Briand, V. R. Brasili, and C. J. Hetmanski, "Developing interpretable models with optimized set reduction for identifying high-risk software components," *IEEE Trans. Softw. Eng.*, vol. 19, no. 11, pp. 1028–1044, Nov. 1993.

[34] S. Kanmani, V. R. Uthariaraj, V. Sankaranarayanan, and P. Thambidurai, "Object-oriented software fault prediction using neural networks," *Inf. Softw. Technol.*, vol. 49, no. 5, pp. 483–492, May 2007.

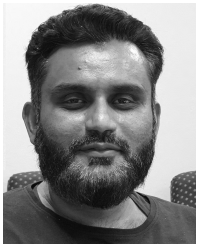
[35] Y. Sasaki, "The truth of the F-measure," *Teach Tutor Mater*, vol. 1, no. 5, pp. 1–5, 2007.

[36] A. Marcus, D. Poshyvanyk, and R. Ferenc, "Using the conceptual cohesion of classes for fault prediction in object-oriented systems," *IEEE Trans. Softw. Eng.*, vol. 34, no. 2, pp. 287–300, Mar. 2008.
[37] A. E. Hassan and R. C. Holt, "The top ten list: Dynamic fault prediction," in *Proc. 21st IEEE Int. Conf. Softw. Maintenance (ICSM)*, Sep. 2005, pp. 263–272.
[38] T. Menzies, J. D. Stefano, K. Ammar, K. McGill, P. Callis, J. Davis, and R. Chapman, "When can we test less?," in *Proc. 5th Int. Workshop Enterprise Netw. Comput. Healthcare Ind.*, Sep. 2003, pp. 98–110.
[39] O. Vandecruys, D. Martens, B. Baesens, C. Mues, M. De Backer, and R. Haesen, "Mining software repositories for comprehensible software fault prediction models," *J. Syst. Softw.*, vol. 81, no. 5, pp. 823–839, May 2008.
[40] N. Chinchor, "MUC-4 evaluation metrics," in *Proc. 4th Conf. Message Understand. (MUC4)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 1992, pp. 22–29. doi: 10.3115/1072064.1072067.
[41] A. Tosun, B. Turhan, and A. Bener, "Validation of network measures as indicators of defective modules in software systems," in *Proc. 5th Int. Conf. Predictor Models Softw. Eng. (PROMISE)*. New York, NY, USA: ACM, 2009, pp. 5:1–5:9.
[42] C. Catal, B. Diri, and B. Ozumut, "An artificial immune system approach for fault prediction in object-oriented software," in *Proc. 2nd Int. Conf. Dependability Comput. Syst. (DepCoS-RELCOMEX)*, Jun. 2007, pp. 238–245.
[43] A. Kaur and R. Malhotra, "Application of random forest in predicting fault-prone classes," in *Proc. Int. Conf. Adv. Comput. Theory Eng. (ICACTE)*, Dec. 2008, pp. 37–43.
[44] E. Arisholm, L. C. Briand, and M. Fuglerud, "Data mining techniques for building fault-proneness models in telecom Java software," in *Proc. 18th IEEE Int. Symp. Softw. Rel. (ISSRE)*, Nov. 2007, pp. 215–224.
[45] A. G. Koru and H. Liu, "Building effective defect-prediction models in practice," *IEEE Softw.*, vol. 22, no. 6, pp. 23–29, Nov. 2005.
[46] S. Kim, E. J. Whitehead, Jr., and Y. Zhang, "Classifying software changes: Clean or buggy?" *IEEE Trans. Softw. Eng.*, vol. 34, no. 2, pp. 181–196, Mar./Apr. 2008.
[47] Y. Singh, A. Kaur, and R. Malhotra, "Software fault proneness prediction using support vector machines," in *Proc. World Congr. Eng.*, vol. 1, 2009, pp. 1–3.
[48] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[49] M. A. de Almeida and S. Matwin, "Machine learning method for software quality model building," in *Proc. Int. Symp. Methodol. Intell. Syst.* Berlin, Germany: Springer, 1999, pp. 565–573.
[50] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald, "Problems with precision: A response to 'Comments on 'Data mining static code attributes to learn defect predictors,'" *IEEE Trans. Softw. Eng.*, vol. 33, no. 9, pp. 637–640, Sep. 2007.
[51] G. J. Pai and J. B. Dugan, "Empirical analysis of software fault content and fault proneness using Bayesian methods," *IEEE Trans. Softw. Eng.*, vol. 33, no. 10, pp. 675–686, Oct. 2007.
[52] G. Denaro, L. Lavazza, and M. Pezzè, "An empirical evaluation of object oriented metrics in industrial setting," *J. Object Technol.*, Sep. 2015.
[53] T. Menzies and J. S. Di Stefano, "How good is your blind spot sampling policy," in *Proc. 8th IEEE Int. Symp. High Assurance Syst. Eng.*, Mar. 2004, pp. 129–138.
[54] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Trans. Softw. Eng.*, vol. 31, no. 10, pp. 897–910, Oct. 2005.
[55] A. B. de Carvalho, A. Pozo, and S. R. Vergilio, "A symbolic fault-prediction model based on multiobjective particle swarm optimization," *J. Syst. Softw.*, vol. 83, no. 5, pp. 868–882, May 2010.
[56] X. Xuan, D. Lo, X. Xia, and Y. Tian, "Evaluating defect prediction approaches using a massive set of metrics: An empirical study," in *Proc. 30th Annu. ACM Symp. Appl. Comput.*, 2015, pp. 1644–1647.
[57] Y. Singh, A. Kaur, and R. Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness models," *Softw. Quality J.*, vol. 18, no. 1, pp. 3–35, 2010.
[58] Y. Jiang, B. Cukic, and T. Menzies, "Fault prediction using early lifecycle data," in *Proc. 18th IEEE Int. Symp. Softw. Rel. (ISSRE)*, Nov. 2007, pp. 237–246.
[59] L. Guo, Y. Ma, B. Cukic, and H. Singh, "Robust prediction of fault-proneness by random forests," in *Proc. 15th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Nov. 2004, pp. 417–428.
[60] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," in *Proc. 30th Int. Conf. Softw. Eng.*, May 2008, pp. 181–190.
[61] 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.
[62] W. Afzal, "Using faults-slip-through metric as a predictor of fault-proneness," in *Proc. 17th Asia Pacific Softw. Eng. Conf. (APSEC)*, Nov./Dec. 2010, pp. 414–422.
[63] B. Turhan and A. Bener, "A multivariate analysis of static code attributes for defect prediction," in *Proc. QSIC*, Oct. 2007, pp. 231–237.
[64] B. Turhan and A. Bener, "Software defect prediction: Heuristics for weighted Naïve Bayes," in *Proc. ICSOFT*, 2007, pp. 244–249.
[65] B. Turhan, G. Kocak, and A. Bener, "Software defect prediction using call graph based ranking (CGBR) framework," in *Proc. 34th Euromicro Conf. Softw. Eng. Adv. Appl.*, Sep. 2008, pp. 191–198.
[66] A. B. de Carvalho, A. Pozo, S. Vergilio, and A. Lenz, "Predicting fault proneness of classes trough a multiobjective particle swarm optimization algorithm," in *Proc. 20th IEEE Int. Conf. Tools Artif. Intell.*, Nov. 2008, pp. 387–394.
[67] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Softw. Eng.*, vol. 14, no. 5, pp. 540–578, 2009.
[68] H. Zhang and X. Zhang, "Comments on 'Data mining static code attributes to learn defect predictors,'" *IEEE Trans. Softw. Eng.*, vol. 33, no. 9, pp. 635–637, Sep. 2007.
[69] B. Turhan and A. Bener, "Analysis of Naive Bayes' assumptions on software fault data: An empirical study," *Data Knowl. Eng.*, vol. 68, no. 2, pp. 278–290, 2009.
[70] V. U. B. Challagulla, F. B. Bastani, and I. Yen, "A unified framework for defect data analysis using the MBR technique," in *Proc. 18th IEEE Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2006, pp. 39–46.
[71] H. M. Olague, L. H. Etzkorn, S. Gholston, and S. Quattlebaum, "Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes," *IEEE Trans. Softw. Eng.*, vol. 33, no. 6, pp. 402–419, Jun. 2007.
[72] A. A. Porter and R. W. Selby, "Empirically guided software development using metric-based classification trees," *IEEE Softw.*, vol. 7, no. 2, pp. 46–54, Mar. 1990.
[73] N. J. Pizzi, A. R. Summers, and W. Pedrycz, "Software quality prediction using median-adjusted class labels," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 3, May 2002, pp. 2405–2409.
[74] Q. Wang, B. Yu, and J. Zhu, "Extract rules from software quality prediction model based on neural network," in *Proc. 16th IEEE Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2004, pp. 191–195.
[75] A. Mahaweerawat, P. Sophatsathit, C. Lursinsap, and P. Musilek, "Fault prediction in object-oriented software using neural network techniques," in *Proc. Adv. Virtual Intell. Comput. Center (AVIC)*, Jan. 2004.
[76] T. J. Ostrand, E. J. Weyuker, and R. M. Bell, "Predicting the location and number of faults in large software systems," *IEEE Trans. Softw. Eng.*, vol. 31, no. 4, pp. 340–355, Apr. 2005.
[77] S. Bibi, G. Tsoumakas, I. Stamelos, and I. Vlahavas, "Regression via Classification applied on software defect estimation," *Expert Syst. Appl.*, vol. 34, no. 3, pp. 2091–2101, Apr. 2008.
[78] Z. Li and M. Reformat, "A practical method for the software fault-prediction," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Aug. 2007, pp. 659–666.
[79] A. Mahaweerawat, P. Sophatsathit, and C. Lursinsap, "Adaptive self-organizing map clustering for software fault prediction," in *Proc. 4th Int. Joint Conf. Comput. Sci. Softw. Eng.*, Khon Kaen, Thailand, 2007, pp. 35–41.
[80] P. Tomaszewski, J. Håkansson, H. Grahn, and L. Lundberg, "Statistical models vs. expert estimation for fault prediction in modified code—An industrial case study," *J. Syst. Softw.*, vol. 80, no. 8, pp. 1227–1238, Aug. 2007.
[81] A. G. Koru and H. Liu, "An investigation of the effect of module size on defect prediction using static measures," *SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–5, May 2005.
[82] Y. Ma, L. Guo, and B. Cukic, *A Statistical Framework for the Prediction of Fault-Proneness* (Advances in Machine Learning Applications in Software Engineering). Singapore: Idea Group Publishing, 2006.
[83] Y. Ma, L. Guo, and B. Cukic, *A Statistical Framework for the Prediction of Fault-Proneness* (Advances in Machine Learning Applications in Software Engineering). Pennsylvania, PA, USA: IGI Global, 2007, pp. 237–263.

[84] K. El Emam, W. Melo, and J. C. Machado, "The prediction of faulty classes using object-oriented design metrics," *J. Syst. Softw.*, vol. 56, no. 1, pp. 63–75, Feb. 2001.
[85] K. El Emam, S. Benlarbi, N. Goel, and S. N. Rai, "Comparing case-based reasoning classifiers for predicting high risk software components," *J. Syst. Softw.*, vol. 55, no. 3, pp. 301–320, Jan. 2001.
[86] K. Kaur, A. Kaur, and R. Malhotra, "Alternative methods to rank the impact of object oriented metrics in fault prediction modelling using neural networks," *Int. J. Eng. Appl. Sci.*, vol. 1, pp. 99–104, Jan. 2005.
[87] S. Zhong, T. M. Khoshgoftaar, and N. Seliya, "Unsupervised learning for expert-based software quality estimation," in *Proc. 8th IEEE Int. Symp. High Assurance Syst. Eng.*, Mar. 2004, pp. 149–155.
[88] C. Catal, U. Sevim, and B. Diri, "Clustering and metrics thresholds based software fault prediction of unlabeled program modules," in *Proc. 6th Int. Conf. Inf. Technol., New Generat. (ITNG)*, Apr. 2009, pp. 199–204.
[89] B. Yang, Q. Yin, S. Xu, and P. Guo, "Software quality prediction using affinity propagation algorithm," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IEEE World Congr. Comput. Intell.*, Jun. 2008, pp. 1891–1896.
[90] N. Seliya and T. M. Khoshgoftaar, "Software quality analysis of unlabeled program modules with semisupervised clustering," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 2, pp. 201–211, Mar. 2007.
[91] Y. Liu, T. M. Khoshgoftaar, and N. Seliya, "Evolutionary optimization of software quality modeling with multiple repositories," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 852–864, Nov. 2010.
[92] N. Seliya and T. M. Khoshgoftaar, "Software quality estimation with limited fault data: A semi-supervised learning perspective," *Softw. Quality J.*, vol. 15, no. 3, pp. 327–344, Sep. 2007.
[93] Q. Wang, J. Zhu, and B. Yu, "Feature selection and clustering in software quality prediction," in *Proc. 11th Int. Conf. Eval. Assessment Softw. Eng. (EASE)*. Salford, U.K.: Swinton, 2007, pp. 21–32.
[94] F. Xing, P. Guo, and M. R. Lyu, "A novel method for early software quality prediction based on support vector machine," in *Proc. 16th IEEE Int. Symp. Softw. Rel. Eng. (ISSRE)*, Nov. 2005, pp. 212–222.
[95] T. M. Khoshgoftaar, E. B. Allen, J. P. Hudepohl, and S. J. Aud, "Application of neural networks to software quality modeling of a very large telecommunications system," *IEEE Trans. Neural Netw.*, vol. 8, no. 4, pp. 902–909, Jul. 1997.
[96] T. M. Khoshgoftaar, E. B. Allen, and J. C. Busboom, "Modeling software quality: The software measurement analysis and reliability toolkit," in *Proc. ICTAI*, Nov. 2000, pp. 54–61.
[97] P. Guo and M. R. Lyu, "Software quality prediction using mixture models with EM algorithm," in *Proc. APAQS*, Oct. 2000, pp. 69–78.
[98] N. F. Schneidewind, "Investigation of logistic regression as a discriminant of software quality," in *Proc. 7th Int. Softw. Metrics Symp. (METRICS)*, 2001, pp. 328–337.
[99] T. M. Khoshgoftaar and N. Seliya, "Comparative assessment of software quality classification techniques: An empirical case study," *Empirical Softw. Eng.*, vol. 9, no. 3, pp. 229–257, Sep. 2004.
[100] T. M. Khoshgoftaar and N. Seliya, "Improving usefulness of software quality classification models based on Boolean discriminant functions," in *Proc. 13th Int. Symp. Softw. Rel. Eng.*, Nov. 2002, pp. 221–230.
[101] T. M. Khoshgoftaar and N. Seliya, "Software quality classification modeling using the SPRINT decision tree algorithm," *Int. J. Artif. Intell. Tools*, vol. 12, no. 3, pp. 207–225, 2003.
[102] F. Lanubile, A. Lonigro, and G. Vissagio, "Comparing models for identifying fault-prone software components," in *Proc. SEKE*, 1995, pp. 312–319.
[103] W. W. Cohen and P. Devanbu, "A comparative study of inductive logic programming methods for software fault prediction," in *Proc. ICML*, 1997, pp. 66–74.
[104] N. Ohlsson, M. Zhao, and M. Helander, "Application of multivariate analysis for software fault prediction," *Softw. Qual. J.*, vol. 7, no. 1, pp. 51–66, Mar. 1998.
[105] T. Menzies, B. Turhan, A. Bener, G. Gay, B. Cukic, and Y. Jiang, "Implications of ceiling effects in defect predictors," in *Proc. 4th Int. Workshop Predictor Models Softw. Eng.*, 2008, pp. 47–54.
[106] W. J. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, no. 1, pp. 32–35, 1950.
[107] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation," in *Proc. Australas. Joint Conf. Artif. Intell.* Berlin, Germany: Springer-Verlag, 2006, pp. 1015–1021.
[108] M. Zhong, "An analysis of misclassification rates for decision trees," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., College Eng. Comput. Sci., 2007.
[109] C. X. Ling, J. Huang, and H. Zhang, "AUC: A statistically consistent and more discriminating measure than accuracy," in *Proc. IJCAI*, vol. 3, 2003, pp. 519–524.
[110] Z. Xu, T. M. Khoshgoftaar, and E. B. Allen, "Prediction of software faults using fuzzy nonlinear regression modeling," in *Proc. 5th IEEE Int. Symp. High Assurance Syst. Eng. (HASE)*, Nov. 2000, pp. 281–290.

MUHAMMAD RIZWAN received the M.S. degree in software engineering from Riphah International University, Islamabad. His master's thesis was on software fault tolerance. He is currently pursuing the Ph.D. degree in computer sciences from the Capital University of Science and Technology (CUST), Islamabad, Pakistan, where he is also a Research Scholar. His research interests include machine learning, software engineering, and software fault tolerance. He is also working on the evaluation of coupling metrics on software fault prediction.

AAMER NADEEM received the M.Sc. degree in computer science from Quaid-i-Azam University (QAU), the M.S. degree in software engineering from the National University of Sciences and Technology (NUST), and the Ph.D. degree in computer science from Mohammad Ali Jinnah University (MAJU). During his Ph.D. degree, he was a Visiting Scholar with The Chinese University of Hong Kong (CUHK) under a research collaboration. He has over 30 years of teaching, research, and industrial experience in computer science and software engineering. He is currently the Head of the Software Engineering Program at the Capital University of Science and Technology (CUST). He is also the Head of the Center for Software Dependability (CSD), a research group at CUST, working in the areas of software reliability, software fault tolerance, formal methods, and safety-critical systems. He has supervised 46 master's and two Ph.D. research theses in the areas of software testing, fault tolerance, and formal methods. He is also an Approved Ph.D. Supervisor for scholars funded by the indigenous fellowship schemes of the Higher Education Commission (HEC) of Pakistan. He has authored or coauthored over 90 papers in reputable international journals and conferences. He is also a Reviewer or an Editorial Board Member of several international peer-reviewed journals and conferences. He is also a Professional Member of the Association for Computing Machinery (ACM).



MUDDASSAR AZAM SINDHU received the M.Sc. degree in computer science from the University of the Punjab, Lahore, Pakistan, and the Licentiate degree in engineering and the Ph.D. degree in computer science from the Royal Institute of Technology (KTH), Sweden. He is currently an Assistant Professor with Quaid-i-Azam University (QAU), Islamabad. He is involved in developing new theories, algorithms, and tools for software testing with a focus on automatic test case generation. More specifically, he is interested in software testing, model mining, algorithms, and formal methods. He is also involved with numerous funded projects. He has authored or coauthored over 12 papers in reputable international journals and conferences.