IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 19 November 2022, accepted 19 December 2022, date of publication 26 December 2022,
date of current version 30 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3232144

RESEARCH ARTICLE

Neg/pos-Normalized Accuracy Measures for Software Defect Prediction

MAOHUA GAN, ZEYNEP YÜCEL, (Member, IEEE), AND AKITO MONDEN, (Member, IEEE)
Graduate School of Natural Science and Technology, Okayama University, Okayama 700-8530, Japan

Corresponding author: Akito Monden (monden@okayama-u.ac.jp)

This work was supported by JSPS KAKENHI under Grant JP20K11749 and Grant JP20H05706.

ABSTRACT In evaluating the performance of software defect prediction models, accuracy measures such as precision and recall are commonly used. However, most of these measures are affected by *neg/pos ratio* of the data set being predicted, where neg is the number of negative cases (defect-free modules) and pos is the number of positive cases (defective modules). Thus, it is not fair to compare such values across different data sets with different neg/pos ratios and it may even lead to misleading or contradicting conclusions. The objective of this study is to address the class imbalance issue in assessing performance of defect prediction models. The proposed method relies on computation of *expected values* of accuracy measures based solely on the value of the neg and pos values of the data set. Based on the expected values, we derive the *neg/pos-normalized accuracy measures*, which are defined as their divergence from the expected value divided by the standard deviation of all possible prediction outcomes. The proposed measures enable us to provide a ranking of predictions across different data sets, which can distinguish between successful predictions and unsuccessful predictions. Our results derived from a case study of defect prediction based on 19 defect data sets indicate that ranking of predictions is significantly different than the ranking of conventional accuracy measures such as precision and recall as well as composite measures F1-value, AUC of ROC, MCC, G-mean and Balance. In addition, we conclude that MCC attains a better defect prediction accuracy than F1-value, AUC of ROC, G-mean and Balance.

INDEX TERMS Software defect, defect prediction model, accuracy measure, classification technology, empirical software engineering.

I. INTRODUCTION AND MOTIVATION

To improve the efficiency of software testing and/or maintenance, it is important to predict defect-prone modules and assign more effort to them. To this end, defect-prone module prediction (or simply, defect prediction) has been studied over decades by many researchers [1], [2], [3], [4], [5], [6], [7], [8], [9], [10].

In assessing the performance of prediction models, it is common to use Precision, Recall, F1-value, Balance [11], Area Under the Curve (AUC) of Receiver Operating Characteristics (ROC) [12], [13], etc. However, there is no consensus on how to interpret these accuracy values. For example, a model with a precision of 0.8 seems to achieve a pretty good performance, but can we confidently claim that this model is successful? The decision will depend on the data set. If the data set contains many defects (e.g. 80% of the modules are defect-prone), a precision of 0.8 is not sufficient for calling that prediction model successful. Namely, we can approach such an accuracy even by assigning classes randomly, whereas, this is not the case for data sets with very few defects. In that respect, the susceptibility of the accuracy measures to the proportion of defect-prone and defect-free modules in the data set is a serious problem [14].

In order to address this issue, we propose deploying the *expected values* of accuracy measures, which we show to be possible to compute solely from the number of positive and negative instances in the data set. In this way, if the (raw) value of a certain accuracy measure is larger than its expected value, the relating model is considered to be a successful one and a proper action is suggested to be taken according to its outcomes.

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

We deploy these expected values in defining *neg/pos-normalized values* of various accuracy measures, which enables us to rank predictions across different data sets. Accounting for neg/pos ratios of the data sets, we can assess the performance of the prediction model under investigation in a more objective way.

In order to demonstrate the capabilities of the proposed approach, we carry out a case study on 38 releases of 19 open source software (OSS) project data sets from 3 data sources. Specifically, we first carry out defect prediction on these sets and evaluate performance in terms of several conventional accuracy measures. We then compare them with their corresponding expected values. The case study confirms that the predictions, which yield higher accuracy than respective expected values are indeed successful.

The rest of the article is organized as follows. In Section II, we elaborate on existing accuracy measures pointing out their shortcomings. We also justify the reason for establishing baseline values for accuracy measures through a preliminary analysis of 19 defect prediction data sets. In Section III we first introduce the basic idea of the proposed approach and then provide explicit definitions. Subsequently, in Section IV we present a case study of defect prediction to demonstrate how well the proposed measures address the susceptibility of the accuracy measures to the proportion of defect-prone and defect-free modules in the data set. Subsequently, in Section V we provide threats to the validity of our work and summarize our main results, contributions and future work in Section VI.

II. BACKGROUND

In binary classification tasks, it is common to evaluate performance based on a contingency table like the one illustrated in Table 1. Specifically, the two dimensions depict the actual and predicted states of the elements of the data set. Here, we use A^+ and A^- to denote actual positives and actual negatives, and P^+ and P^- to denote predicted positives and predicted negatives. As framed within the scope of this study, an *actual positive* is a defect-prone module, whereas an *actual negative* is a defect-free module.

TABLE 1. Contingency table for binary classification tasks.

		Predicted state	
		P^+	P^-
Actual state	A^+	TP	FN
	A^-	FP	TN

The prediction results that correctly indicate the presence or absence of a defect are referred to as *True Positive (TP)* and *True Negative (TN)*, respectively. In addition, the prediction results, which wrongly indicate the presence of a defect, are called *False Positive (FP)* and the ones, which wrongly indicate the absence of a defect, are called *False Negative (FN)*. In Table 1 the number of such prediction results are denoted with TP, TN, FP, FN in the corresponding cells.[1]

[1]Note that in Table 1 $TP, TN, FP, FN \in \mathbf{Z}_{\geq 0}$.

Based on such values, one can calculate various measures to evaluate classification performance. Some conventional measures involve Precision, Recall, NPV (negative predictive value) and Specificity.[2] Henceforth, we use the normal font to refer to an accuracy measure (e.g. Precision) and typewriter font when it is treated as a random variable or a value that it takes (e.g. `Precision` and `precision`, respectively).

In terms of the entries in Table 1, the above-mentioned conventional accuracy measures are computed explicitly as:

$$\texttt{precision} = \frac{TP}{TP + FP}; \tag{1}$$

$$\texttt{recall} = \frac{TP}{TP + FN}; \tag{2}$$

$$\texttt{npv} = \frac{TN}{FN + TN}; \tag{3}$$

$$\texttt{specificity} = \frac{TN}{TN + FP}. \tag{4}$$

A. WEAKNESS OF CONVENTIONAL ACCURACY MEASURES

It is common to use Precision and Recall for evaluating performance in the detection of positive cases (i.e. defect-prone modules) [2], [6], [15], whereas NPV and Specificity are used mostly for evaluating performance concerning the identification of negative cases (i.e. defect-free modules) [16], [17], [18], [19], [20], [21]. Nevertheless, there is no universal agreement on which measure is the most appropriate one for each detection task.

For various purposes such as quality assurance, Precision and Recall are the most intuitive measures, since they reflect performance in detecting positive cases (i.e. defect-prone modules). Nevertheless, as discussed by Zhang and Zhang [22], a coupled deliberation of Precision and Recall is indispensable, since there is an inherent trade-off between them. Namely, in order to increase `precision`, one may simply consider as positive only those cases with a very high probability of being positive. But this will raise *FN* and thus decrease `recall`. On the other hand, if the condition for being considered positive is loosened, `recall` can be improved, at the cost of deteriorating `precision`. However, even a coupled deliberation may not be sufficient in certain cases. For instance, Chicco and Jurman [24] argue that the F1-value (which is the harmonic mean of the `precision` and `recall`) can dangerously show overoptimistic inflated results, especially on imbalanced data sets.

Therefore, in addition to the detection of positive cases (i.e. Precision and Recall), one needs to pay regard also to the detection of negative cases [23], [24]. Indeed, a `recall` of 1.0 can be easily achieved by predicting all modules as positive, but this would result in poor predictive performance for negative cases, i.e. `specificity` becomes zero. In this respect, Chicco and Jurman [24] recommend using Matthews Correlation Coefficient (MCC), which considers both positive and negative prediction performance.

[2]Precision is also known as Positive Predictive Value, whereas Recall and Specificity are also referred to as True Positive Rate and True Negative Rate, respectively.

However, note that an important criticism to all such performance measures is their susceptibility to certain intrinsic features of the data. In defect prediction domain, Menzies et al. [14] argue that the accuracy measures in Equations 1∼4 as well as MCC are not robust [26], [27], [28], since they are very often affected by the *neg/pos* ratio of the data set, where *pos* is the number of actual positive (i.e. defect-prone) modules (i.e. A^+) and *neg* is the number of actual negative (i.e. defect-free) modules (i.e. A^-),

$$\text{Neg/pos ratio} = \frac{A^-}{A^+}. \tag{5}$$

For instance, a large neg/pos ratio tends to yield low `precision`. Nevertheless, Menzies et al. also argue that a low `precision` per se is not always a problem, since there are many cases where a low `precision` is acceptable. On the other hand, although composite measures such as AUC of ROC, Balance and G-mean are known to be more robust against neg/pos ratio [25], [47], [48], according to their definition, AUC of ROC and Balance focus solely on positive cases and ignore the negative ones, whereas G-mean [7] ignores the precision of both positive and negative cases.

In this paper, exploiting the intuitive nature of four conventional accuracy measures of Precision, Recall, NPV and Specificity, we propose using their expected values, which can be regarded as a baseline, to overcome their susceptibility to neg/pos ratio, and to avoid over-optimistic or over-pessimistic interpretations. We also propose a neg/pos-normalization scheme to enable a fair and objective assessment of model performance, and define the "successful prediction" based on proposed neg/pos-normalization scheme to focus on both positive and negative cases, and both recall and precision. By comparing prediction results across different data sets with different neg/pos ratios, we show that the neg/pos-normalized values are more robust against the composition of the data set. We compare our measures with the F1-value, AUC of ROC, MCC, G-mean and Balance to show how these metrics give a similar or different evaluation for the same model, and show that there are some cases where these composite measures are not considered useful based on the definition of "successful prediction".

B. PRELIMINARY ANALYSIS

To demonstrate the influence of neg/pos ratio on the assessment of defect prediction performance, we conduct a preliminary analysis.

1) DATA SETS AND PREDICTION METHOD

We collected 19 open source software project data sets from 3 sources (see Table 2). Specifically, the data sets from no.1 to no.4 (MYLN, PDE, JDT and NBNS) are introduced by Kamei et al. and the details on data collection and measurement methods can be found in [8], [29], and [30]. The data sets from no. 5 to no.13 (ANT, CAML, FRST, JEDT, LOG4, LUCN, POI, PROP, SYNP) are donated by Jureczko and Madeyski [3], [4] to the SeaCraft repository [33] and their specifics are reported in [4]. The remaining data sets from no.14 to no.19 (ECOS, EXIM, GNY, HLMA, HBNT, XDOC) are collected from the Software Engineering Data Repository for Research and Education [31] and their details are elaborated on in [32].[3]

Note that each data set in Table 2 is represented with 2 releases, which enables a cross-version defect prediction. In our preliminary experiments, we performed such defect prediction, where the older release is used as training data and the newer release is used as test data. As for the specific prediction method, we deployed random forest, since it is shown by Lessmann et al. to constitute one of the best-performing defect prediction methods [9]. We used the library "randomForest" of R and its default hyperparameter (the number of tree = 500).

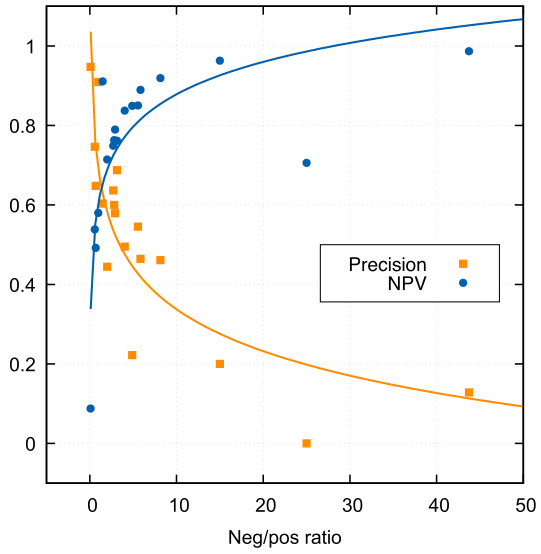

FIGURE 1. Relationship between neg/pos ratio and precision (r=-0.758) and NPV (r=0.469) in cross-version defect prediction (see Equations 1 and 3). Dashed curves represent approximations with logarithmic function and r denotes the correlation coefficient.

2) RESULTS OF THE PRELIMINARY ANALYSIS

The results of our preliminary analysis are illustrated in Figure 1, where the x-axis shows the neg/pos ratio (see Equation 5) and the y-axis depicts the `precision` or `npv` of cross-version defect prediction.[4] From Figure 1, we can see clearly that `precision` decreases with increasing neg/pos ratio, whereas `npv` increases. In other words, concerning data sets with a high number of positives (i.e. defect-prone modules), `precision` may turn out to be high as a natural result of the composition of the set. Similarly, regarding data sets with a high number of negatives (i.e. defect-free modules), `npv` may unfold to be higher, merely due to the simplification of detection (of negatives).

[3]Note that from these 19 datasets we removed those modules with zero lines of code and used the remaining modules in our analysis.

[4]In order to have a clear illustration, we chose two out of the four conventional accuracy measures in Equation 1∼4, where one focuses on positive cases and the other on negative cases.

TABLE 2. Summary of the defect data sets used in the experiments. The maximum and minimum rate of defect-prone modules are indicated in bold face with *.

No.	Project Name	Short Name	Version	Number of total modules T	Number of defect-prone modules A^+	Percentage of defect-prone modules	Neg/pos ratio A^-/A^+
1	Mylyn	MYLN	2	1230	848	68.9	0.450
			3	1502	606	40.4	1.479
2	Eclipse	PDE	3.1	228	49	21.5	3.653
			3.2	309	79	25.6	2.911
3	Eclipse	JDT	3.1	3192	528	16.5	5.045
			3.2	3408	373	10.9	8.137
4	Netbeans	NBNS	4	4660	898	19.3	4.189
			5	9332	1365	14.6	5.837
5	Ant	ANT	1.5	293	32	10.9	8.156
			1.6	350	92	26.3	2.804
6	Camel	CAML	1.4	856	144	16.8	4.944
			1.6	945	188	19.9	4.027
7	Forrest	FRST	0.7	29	5	17.2	4.800
			0.8	32	2	6.3	15.000
8	Jedit	JEDT	4.2	367	48	13.1	6.646
			4.3	492	11	2.2	43.727
9	Log4j	LOG4	1.1	109	37	33.9	1.946
			1.2	205	189	**92.2***	**0.085***
10	Lucene	LUCN	2.2	247	144	58.3	0.715
			2.4	340	203	59.7	0.675
11	Poi	POI	2.5	384	248	64.6	0.548
			3	441	281	63.7	0.569
12	Prop	PROP	4	8702	840	9.7	9.360
			5	8506	1298	15.3	5.553
13	Synapse	SYNP	1	157	16	10.2	8.813
			1.1	222	60	27	2.700
14	eCos	ECOS	2	621	110	17.7	4.645
			3	3459	67	**1.9***	**50.627***
15	Exim	EXIM	4.63	184	28	15.2	5.571
			4.68	61	31	50.8	0.968
16	Ganymede	GNY	1.0pre	99	29	29.3	2.414
			1	90	30	33.3	2.000
17	Helma	HLMA	1.3.1	145	38	26.2	2.816
			1.4.0	100	17	17	4.882
18	Hibernate	HBNT	3.6.1	374	87	23.3	3.299
			3.6.2	4878	1178	24.1	3.141
19	XDoclet	XDOC	1.2Beta2	130	5	3.8	25.000
			1.2Beta3	102	30	29.4	2.400

As a result, we can say that `Precision` and `Npv` depend considerably on the composition (i.e. neg/pos ratio) of the particular data set under investigation. In that respect, judging the performance of a model based on (raw) accuracy measures without paying regard to their neg/pos ratios may lead to misleading or contradicting conclusions.

III. PROPOSED MEASURES

Our basic assumption is that all possible prediction outcomes with the *correct composition* have the *same likelihood* to occur. This assumption is determined paying regard to (i) fairness and (ii) desirability of prediction.

Fairness addresses the distribution of positives and negatives in the data set and the impact of that on accuracy metrics. Consider defect prediction on a data set, where actual class of all modules are positive (i.e. defect-prone). In that case, the only possible `precision` is 1.0. It is not fair to compare the performance of some model on this data set to its performance on another data set. Similarly, concerning defect prediction on a data set with all negatives (i.e. defect-free modules), the only possible `precision` is 0.0, which again precludes a fair comparison to another data set.

Desirability addresses the condition, in which the percentage of defect-prone modules in prediction should match the percentage in the test data. In practice, this is not easy to achieve, and overestimation and underestimation frequently occur due to the difference in the percentage of defect-prone modules in training and test data. To address this issue, researchers propose remedies based on numerical approaches, such as misclassification rate balancing [34], over-/under-sampling [1], [35], [36], [37], and transfer-learning [38], [39].

To satisfy fairness and desirability and to evaluate prediction models in an unprejudiced way, we believe that it is necessary to account for the proportion of defect-free and defect-prone modules in the data set. To that end, in what follows, we first discuss the expected values of accuracy measures and then introduce the proposed normalization scheme.

A. EXPECTED VALUES OF ACCURACY MEASURES

In this section, we first compute of expected values of *TP*, *TN* etc., deploy them to compute the expected values of conventional accuracy measures, explaining relating specifics on a general case, whereas a demonstration on a specific toy example is provided in Appendix A.

As mentioned in Section III, our basic assumption is that all possible prediction outcomes with the correct composition occur with the same likelihood. Suppose that we perform defect prediction on a test data set with T modules. Let the number of actual positives and actual negatives be denoted with A^+ and A^-, respectively (see also Table 1). Clearly, $T = A^+ + A^-$ for actual states, and also $T = P^+ + P^-$ for predicted states.

Let an arbitrary prediction result[5] be represented with a vector π_i and the index i denotes the i-th possible prediction result. Since there are T modules in the test data set, we may use a binary vector with T components (i.e. using a "1" for a predicted positive module and a "0" for a predicted negative module.).

Predicating on fairness and desirability, we consider only those vectors π_i for which $P^+ = A^+$ and $P^- = A^-$. In other words, A^+ modules are predicted as positive and A^- modules are predicted as negative, all of which are not necessarily predicted correctly. Suppose that $\Pi(A^+, A-) = \{\pi_i\}$ is the set of all such vectors (i.e. prediction results), Specifically, the elements of $\Pi(A^+, A^-)$ are permutations of A^+ "1"s and A^- "0"s.

The number of elements of $\Pi(A^+, A^-)$ is the total number of all such permutations. Namely,

$$\#\Big(\Pi(A^+, A^-)\Big) = \binom{T}{A^+}, \tag{6}$$

as $\#(\cdot)$ denotes the number of elements of a set and $T = A^+ + A^-$ as mentioned above.

Let us focus on a particular actual positive module. There will be $\#\Big(\Pi(A^+, A^-)\Big)$ predictions for this module (i.e. one in each vector π_i). In addition, due to our basic assumption, A^+/T of those predictions will indeed be positive (i.e. True Positive). Thus, concerning each actual positive case, the number of positive predictions in all vectors π_i is given by

$$\frac{A^+}{T} \cdot \binom{T}{A^+}. \tag{7}$$

Moreover, since there are A^+ actual positive modules in the test data, the total number of positive modules, which are correctly predicted as positive (i.e. *TP*), is

$$TP = A^+ \cdot \frac{A^+}{T} \cdot \binom{T}{A^+}. \tag{8}$$

From this value, the expected value of *TP* can be computed as *TP per prediction*. Remember that the number of predictions (i.e. possible permutations) is as given in Equation 6. Thus, we get

$$\begin{aligned}\mathbf{E}(TP) &= \frac{TP}{\#\Big(\Pi(A^+, A^-)\Big)} = \frac{\frac{A^+ \cdot A^+ \cdot \binom{T}{A^+}}{T}}{\binom{T}{A^+}} \\ &= \frac{(A^+)^2}{T}. \end{aligned} \tag{9}$$

Also, the total number of *FP*, *FN* and *TN* can be computed in a similar way to Equation 8 and their respective expected values can be obtained in a similar way to Equation 9.[6]

Finally, such expected values can be deployed in computing the expected values of the commonly used accuracy measures. In particular, the expected values for the measures given in Equations 1∼4 can be computed as

$$\mathbf{E}(\texttt{Precision}) = \frac{\mathbf{E}(TP)}{\mathbf{E}(TP) + \mathbf{E}(FP)} = \frac{A^+}{T}, \tag{10}$$

$$\mathbf{E}(\texttt{Recall}) = \frac{\mathbf{E}(TP)}{\mathbf{E}(TP) + \mathbf{E}(FN)} = \frac{A^+}{T}, \tag{11}$$

$$\mathbf{E}(\texttt{Npv}) = \frac{\mathbf{E}(TN)}{\mathbf{E}(FN) + \mathbf{E}(TN)} = \frac{A^-}{T}, \tag{12}$$

$$\mathbf{E}(\texttt{Specificity}) = \frac{\mathbf{E}(TN)}{\mathbf{E}(TN) + \mathbf{E}(FP)} = \frac{A^-}{T}. \tag{13}$$

Interestingly, expected values for the accuracy measures concerning the identification of positive cases (i.e. `Precision` and `Recall`) are found to be A^+/T, and those relating to negative cases (i.e. `Npv` and `Specificity` are found to be A^-/T. This indicates that, provided that a test data set contains many positive instances (i.e. defect-prone modules), then high `precision` and `recall` are essential, while low `npv` and `specificity` are acceptable.

B. NEG/POS-NORMALIZED ACCURACY MEASURES

To demonstrate how to use expected values in interpreting the values of accuracy measures concerning a certain prediction model, we give a hypothetical example in Figure 2. Let M denote an accuracy measure. Suppose that running defect prediction on three data sets A, B, C with a certain defect prediction model yields the accuracy values `m`, which are demonstrated in Figure 2 together with their expected values $\mathbf{E}(\texttt{M})$.

If we interpret prediction performance based only on `m`, the prediction for data set C seems much better than those for A and B. However, note that concerning data set C the expected value of `M` is even higher than its empirical value, i.e. $\mathbf{E}(\texttt{M}) > \texttt{m}$. In other words, if we keep making random predictions for C, eventually we will get better accuracy than `m`. In this sense, we consider a prediction to be successful, only if the accuracy measures yield better numbers than their expected values.

In addition to this simple binary (better or worse) assessment, one can also quantify *how much better or worse* the prediction model is than the expected values. Of course, it

[5]Here, the "arbitrariness" does not refer to the stochasticity of the prediction model. It simply refers to the fact that one can apply this method to any data set with a known neg/pos ratio, assuming equal probabilities for each possible prediction outcome delivered by a (black box) prediction model.

[6]Specifically, $\mathbf{E}(FP)$, $\mathbf{E}(FN)$ and $\mathbf{E}(TN)$ are respectively A^+A^-/T, A^+A^-/T and $(A^-)^2/T$.

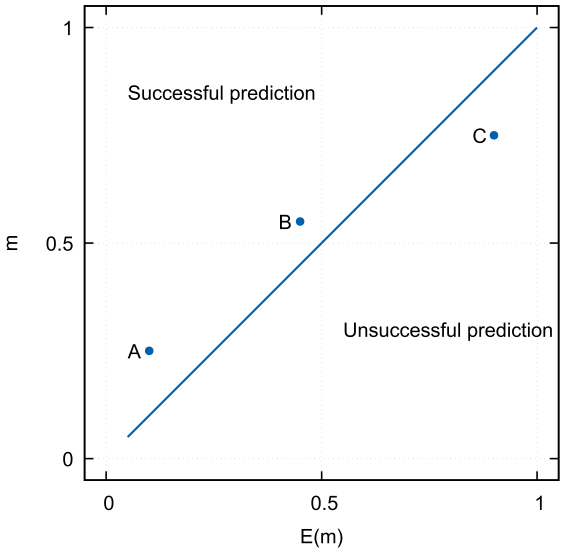

FIGURE 2. Accuracy values $\mathtt{m}$ and expected values $\mathbf{E}(\mathtt{M})$ for three data sets.

is plausible to consider that the higher is the empirical value than the expected value, the better is the prediction performance. However, taking the difference between the empirical and expected values straightforwardly is not adequate, since one needs to consider also the variance in possible predictions. In that respect, concerning an accuracy value $\mathtt{m}$, we define the *neg/pos-normalized value* $\bar{\mathtt{m}}$ as follows:

$$\bar{\mathtt{m}} = \frac{\mathtt{m} - \mathbf{E}(\mathtt{M})}{\sigma_\pi} \tag{14}$$

where σ_π denotes the standard deviation of all possible prediction outcomes. Specifically,

$$\sigma_\pi = \sqrt{\frac{\sum_{i=1}^{\#(\Pi)} (\pi_i - \mathbf{E}(\Pi))^2}{\#(\Pi)}} \tag{15}$$

where $\mathbf{E}(\Pi)$ denotes the expected value of π_i (see also Appendix A).[7]

Using the neg/pos-normalized measures, we can assess how much better or worse the prediction results are than the expected values. For example, assuming one normal distribution based on all possible predictions, if the neg/pos-normalized `precision`, i.e. $\overline{\texttt{precision}}$, is larger than 0, then the prediction is better than 50% of all possible predictions, and thus it can be considered as successful. On the other hand, if $\overline{\texttt{precision}} > 1$, then the prediction is 1σ better than an average prediction and so it is better than 84% of all possible predictions, which is quite successful.

In this paper, a prediction for which neg/pos-normalized recall, precision, specificity, NPV are all positive is considered to be a successful prediction, otherwise it is an unsuccessful prediction.

[7]Note that in Equation 15, $\#(\Pi)$ is simply a shorthand notation for $\#\left(\Pi(A^{-} \cdot A^{+})\right)$ appearing in Equation 6.

IV. CASE STUDY

The purpose of this case study is to demonstrate that large values of accuracy measures do not necessarily indicate successful predictions and vice versa. For drawing attention to the dangers of using conventional accuracy measures in comparing prediction results across different data sets, we contrast the rankings of data sets in terms of raw and neg/pos normalized accuracy values.[8]

A. DATA AND METHODOLOGY

As data, we use the same sets deployed in the preliminary analysis (see Table 2). Similar to Section II-B, we conduct a cross-version defect prediction, where the old version of each project is used in training and its new version is used in testing. As a defect prediction model, we employ random forest, since it was shown to be one of the best models in the defect prediction domain and in other classification problems in terms of prediction performance and stability [9], [40], [41], [42], [43].

In addition, we evaluate commonly-used composite measures F1-value, AUC of ROC, MCC, G-mean and Balance by comparing their ranking results with our ranking by neg/pos-normalized measures.

Figure 3 shows the procedure of our case study. Firstly, we use training data to conduct model construction. Secondly, we make predictions on the test data and calculate raw values of accuracy measures. Thirdly, we calculate their expected values (e.g. as in Equation 10) and the standard deviations (see Equation 15). Finally, we calculate corresponding neg/pos-normalized values as in Equation 14.[9]

In this study, we perform 30 repetitions of the prediction experiment, and take the average of them as the value of the prediction experiment.

B. COMPARISON OF ACCURACY MEASURES TO THEIR EXPECTED VALUES

Figure 4 and Table 3 show the relationship between raw (empirical) values of the four conventional accuracy measures as well as their expected and neg/pos normalized values.[10] Figure 4 provides the overall distribution of raw and expected values and Table 3 lists the values accurately.

Regarding positive accuracy measures, we first examine `Precision` in Figure 4-(a) and `Recall` in Figure 4-(b).[11]

In Figure 4-(a), the data point at the top right corner represents data set no.9 (LOG4), which attains a remarkably

[8]Note that producing a ranking of data sets according to the accuracy values of some prediction model is not one of our purposes. Nevertheless, we consider it as a simple and clear way to demonstrate the discrepancy between raw and neg/pos normalized accuracy values.

[9]While computing raw values of some accuracy measures, there are cases where the denominator becomes zero because of too large neg/pos ratio (e.g. data set no. 19 XDOC). In our experiment, we use $\bar{m} = 0$ for such cases.

[10]In Figure 4, each color denotes a different data set. The gradation of colors does not imply any progressive relationship between data sets.

[11]In Figure 4-(a), there is one unsuccessful prediction. Namely, for data set no.19 (XDOC) `Precision` $= 0$, since a valid prediction model could not be constructed due to too few instances of positive cases in training data (see also Table 2).

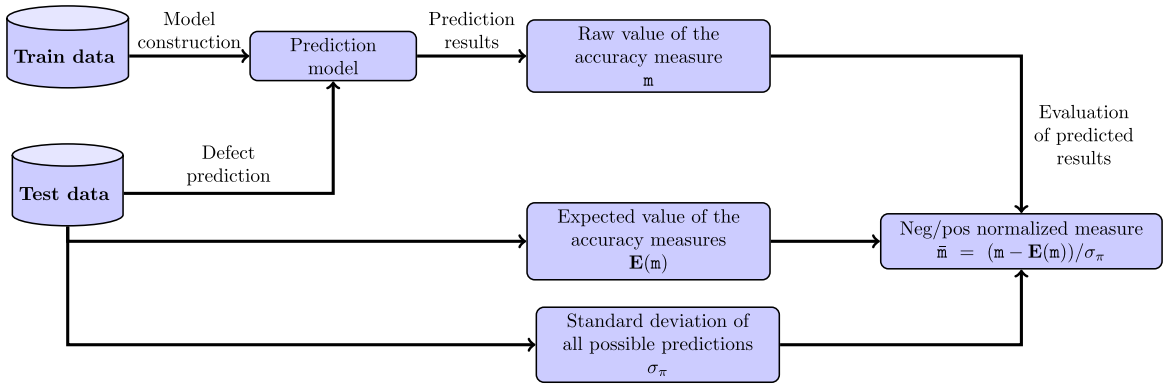

FIGURE 3. Procedure of cross-version experiment.

TABLE 3. Values of conventional accuracy measures and their neg/pos-normalized values for each data set.

		(a) Raw accuracy values				(b) Expected accuracy values				(c) Neg/pos-normalized accuracy values			
No	Project	prec.	recall	npv	specif.	E(Prec.)	E(Recall)	E(Npv)	E(Specif.)	$\overline{\text{prec.}}$	$\overline{\text{recall}}$	$\overline{\text{npv}}$	$\overline{\text{specif.}}$
1	MYLN	0.604	0.916	0.913	0.594	0.403	0.403	0.597	0.597	0.497	1.269	0.439	-0.004
2	PDE	0.547	0.257	0.784	0.927	0.256	0.256	0.744	0.744	1.124	0.005	0.051	0.232
3	JDT	0.467	0.325	0.920	0.954	0.109	0.109	0.891	0.891	3.233	1.947	0.033	0.071
4	NBNS	0.462	0.324	0.890	0.936	0.146	0.146	0.854	0.854	2.159	1.211	0.042	0.094
5	ANT	0.606	0.160	0.763	0.963	0.263	0.263	0.737	0.737	1.292	-0.388	0.033	0.288
6	CAML	0.486	0.268	0.836	0.930	0.199	0.199	0.801	0.801	1.430	0.344	0.043	0.156
7	FRST	0.200	0.500	0.963	0.867	0.063	0.063	0.938	0.938	0.788	2.506	0.027	-0.075
8	JEDT	0.128	0.455	0.987	0.929	0.022	0.022	0.978	0.978	2.127	8.738	0.009	-0.050
9	LOG4	0.947	0.282	0.087	0.813	0.922	0.922	0.078	0.078	0.027	-0.694	0.010	0.794
10	LUCN	0.651	0.691	0.496	0.451	0.597	0.597	0.403	0.403	0.090	0.157	0.130	0.067
11	POI	0.746	0.721	0.537	0.568	0.637	0.637	0.363	0.363	0.170	0.132	0.238	0.279
12	PROP	0.552	0.028	0.850	0.996	0.153	0.153	0.847	0.847	2.611	-0.813	0.004	0.173
13	SYNP	0.619	0.106	0.747	0.976	0.270	0.270	0.730	0.730	1.270	-0.600	0.022	0.317
14	ECOS	0.080	0.320	0.986	0.927	0.019	0.019	0.981	0.981	2.390	11.778	0.005	-0.054
15	EXIM	0.895	0.287	0.567	0.966	0.508	0.508	0.492	0.492	0.755	-0.432	0.107	0.670
16	GNY	0.456	0.419	0.721	0.751	0.333	0.333	0.667	0.667	0.361	0.251	0.073	0.113
17	HLMA	0.236	0.335	0.850	0.775	0.170	0.170	0.830	0.830	0.348	0.873	0.024	-0.065
18	HBNT	0.705	0.019	0.761	0.997	0.241	0.241	0.759	0.759	1.916	-0.922	0.004	0.300
19	XDOC	0.000	0.000	0.706	1.000	0.294	0.294	0.706	0.706	-0.973	-0.973	0.000	0.385

high precision. Nevertheless, it is almost the same as its expected value and the neg/pos ratio is considerably low (i.e. 0.085, see also Table 2).[12] Thus, although the precision is high, it is not sufficient to prove the efficacy of the prediction model. Therefore, we check the recall value relating to this data set in Figure 4-(b) (see the right most data point) and see that the raw value is lower than the expected value. We can also confirm In Table 3-(b) that recall = 0.282 and **E**(Recall) = 0.922, which is a significant difference. Hence, although we cannot draw a reliable inference from precision, based on recall, we can say that this prediction is not good. As a matter of fact, since this data set has many defect-prone modules (see Table 2), there is little value in predicting defects, and it is preferable to test just all modules.

On the other hand, data set no.14 (ECOS) attains the lowest precision and yet it is more than three times larger than its expected value (i.e. precision = 0.080, **E**(Precision) = 0.019, see Table 3-(a), (b)). Note that relating to this data set also the recall is much greater than its expected value (i.e. recall = 0.320, **E**(Recall) = 0.019, see Table 3-(a), (b)) and also that the neg/pos ratio is considerably high (i.e. 50.6, see Table 2). Therefore, for data set no.14 the low precision is considered not to imply a bad prediction. Similarly, despite fair values of recall, the predictions relating to data sets no.1 (MYLN) and no.14 (ECOS) are actually successful, since their recall values (and other accuracy measures) are greater than their expected values.

Regarding negative accuracy measures, we examine npv in Figure 4-(c) and specificity in Figure 4-(d).

In Figure 4-(c), npv of all data sets are seen to be very close to their expected values. This indicates that a small/large npv implies a small/large expected value **E**(Npv), which makes it insufficient for the judgment of in/efficiency of prediction performance.

Therefore, we examine the specificity values given in Figure 4-(d). Five data sets (no. 1, 7, 8, 14, 17) are seen to have lower specificity than their expected values

[12]Actually, we can read the exact values relating to data set no.9 as precision = 0.947 and **E**(Precision) = 0.922) in Table 3-(a) and (b).

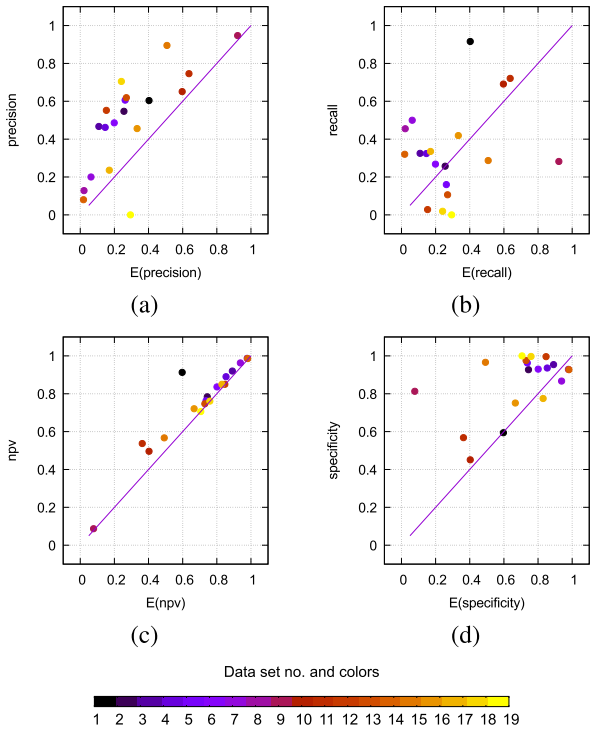

FIGURE 4. Scatter diagram of raw values and expected values. (a) `Precision`, (b) `Recall`, (c) `Npv` and (d) `Specificity`.

(see also Table 3). For example, data set no. 8 (JEDT) has `specificity` = 0.929, which is very high but not as high as its expected value $\mathbf{E}$(`Specificity`) = 0.978. Thus, despite the high `specificity`, we cannot say that the prediction is successful. Note also that the relating neg/pos ratio is 43.727 (see Table 2). On the other hand, data set no. 10 (LUCN) has the lowest `specificity` of 0.451, but it is greater than its expected value $\mathbf{E}$(`Specificity`) = 0.403.

These results indicate the danger of linking the empirical value of (raw) accuracy measures directly to performance evaluation, and also show that misleading inferences can be avoided by the proposed comparison procedure with corresponding expected values.

Note that the probability of detection (*pd*) and the probability of false alarm (*pf*), which are often used in defect prediction studies [11], [14], are considered to be unreliable in some datasets. It is because *pd* and *pf* are referred as "`recall`" and "1-`specificity`" in this study and neg/pos-normalized `recall` and `specificity` are not always reliable as shown in Table 3.

C. PERFORMANCE EVALUATION WITH NEG/POS-NORMALIZED MEASURES

In this section, we first compute the neg/pos-normalized measures (see Equation 14) corresponding to the four conventional accuracy measures given in Equations 1∼4. Then, we provide two performance rankings with respect to the raw and neg/pos normalized values of the accuracy measures

1) RANKING OF PREDICTIONS FOR DIFFERENT DATA SETS

Table 3-(a) and (c) show respectively the raw and neg/pos-normalized values concerning the four conventional accuracy measures.

As it can be seen in Table 3-(c), for several data sets the neg/pos normalized values are negative, especially for `recall` and `specificity`. Notably, data set no. 3 (JDT) has the highest `precision` of 3.233, although the `precision` itself is not very high (i.e. 0.467). On the other hand, data set no. 9 (LOG4) has the lowest `precision` of 0.027, although the `precision` itself is the highest (i.e. 0.947). Thus, it is already evident from these tables that neg/pos normalization leads to a large difference in relative performances.

Subsequently, we rank the predictions with respect to their performance in terms of all accuracy measures. To that end, we use the win-tie-loss method [44], [45] and we aggregate all pairwise win-tie-loss values concerning the 4 accuracy measures into one because they are based on similar principles. Specifically, concerning a particular accuracy measure, if data set A attains a higher value than data set B, the number of *win*s concerning A and the number of *losses* concerning B are incremented by one, whereas for equal values, the number of *tie*s concerning A and B is added by one.[13]

Table 4-(a) shows the ranking of data sets based on conventional (raw) accuracy measures, whereas Table 4-(b) shows the ranking based on the neg/pos-normalized values of those measures. We immediately notice that the two rankings are quite different. Some of the data sets, for which neg/pos normalization leads to a large change in rank, involve CAML (13 → 3), GNY (18 → 11), and JEDT (3 → 10). Taking a closer look at CAML data set, we see that it ranks 11, 13, 9, and 9 with respect to `precision`, `recall`, `npv`, `specificity`, respectively. And, after neg/pos normalization the respective ranks become 7, 8, 7, 10, rising a few degrees on average.

2) COMPARISON WITH CONVENTIONAL COMPOSITE MEASURES

As composite accuracy measures, we use the F1-value, AUC of ROC, MCC, G-mean and Balance since these are commonly used accuracy measures in two-group classification studies (including software defect prediction). We compute their (raw) values (we perform 30 repetitions and take their average as shown in Table 5), rank them according to these values and compare this ranking to the one reported in Table 4.

In addition, we evaluate the parallel between each pair of rankings based on the correlation coefficient (r). Note that a correlation coefficient (in absolute value) smaller than 0.36 is generally considered to indicate a low or weak correlation, 0.36 to 0.68 a modest or moderate correlation, and 0.68 to 1.0 a strong or high correlation [46].

[13]Note that the original win-tie-loss method uses a statistical test to judge win or loss, whereas this paper simply judges based on the difference in accuracy values, since statistical tests (such as Wilcoxon rank-sum test) do not apply to accuracy measures in two-group classification.

TABLE 4. Ranking of predictions for 19 data sets based on (a) conventional (raw) accuracy measures and (b) neg/pos-normalized accuracy measures.

(a)

	Rank	Win	Tie	Loss	Win-Loss
MYLN	1	45	0	27	18
JDT	1	45	0	27	18
JEDT	3	43	0	29	14
EXIM	4	42	0	30	12
FRST	5	40	0	32	8
HBNT	5	40	0	32	8
NBNS	5	40	0	32	8
PROP	8	39	1	32	7
SYNP	9	37	0	35	2
ANT	9	37	0	35	2
POI	11	36	0	36	0
ECOS	12	34	1	37	-3
CAML	13	34	0	38	-4
HLMA	14	31	1	40	-9
LUCN	15	31	0	41	-10
PDE	16	30	1	41	-11
LOG4	17	30	0	42	-12
GNY	18	26	0	46	-20
XDOC	19	22	0	50	-28

(b)

	Rank	Win	Tie	Loss	Win-Loss
JDT	1	48	1	23	25
NBNS	2	46	0	26	20
EXIM	3	44	0	28	16
CAML	3	44	0	28	16
MYLN	5	42	0	30	12
PDE	6	40	0	32	8
POI	6	40	0	32	8
ANT	8	39	1	32	7
ECOS	9	39	0	33	6
JEDT	10	38	0	34	4
GNY	11	37	0	35	2
SYNP	12	35	0	37	-2
FRST	13	32	0	40	-8
LUCN	13	32	0	40	-8
PROP	15	30	1	41	-11
HBNT	16	29	1	42	-13
LOG4	17	27	0	45	-18
HLMA	18	24	0	48	-24
XDOC	19	16	0	56	-40

TABLE 5. Successful prediction and (raw) value of composite measures.

	Successful or not	F1-value	AUC of ROC	MCC	G-mean	Balance
MYLN	NO	0.728	0.857	0.513	0.738	0.707
PDE	YES	0.350	0.671	0.247	0.488	0.472
JDT	YES	0.383	0.749	0.328	0.557	0.521
NBNS	YES	0.381	0.755	0.302	0.550	0.519
ANT	NO	0.253	0.766	0.213	0.392	0.405
CAML	YES	0.345	0.669	0.252	0.499	0.480
FRST	NO	0.286	0.930	0.244	0.658	0.634
JEDT	NO	0.199	0.730	0.209	0.650	0.611
LOG4	NO	0.435	0.551	0.057	0.479	0.476
LUCN	YES	0.670	0.639	0.145	0.558	0.555
POI	YES	0.733	0.659	0.286	0.640	0.636
PROP	NO	0.054	0.619	0.099	0.168	0.313
SYNP	NO	0.180	0.658	0.172	0.320	0.367
ECOS	NO	0.128	0.763	0.128	0.544	0.517
EXIM	NO	0.434	0.571	0.342	0.526	0.495
GNY	YES	0.437	0.657	0.173	0.560	0.553
HLMA	NO	0.276	0.643	0.097	0.508	0.503
HBNT	NO	0.036	0.629	0.087	0.137	0.306
XDOC	NO	0	0.654	0	0	0.293

Figure 5 shows the distribution of pairs of rankings for each composite accuracy measure. In each diagram, the x-axis represents the ranking based on neg/pos-normalized measure (given in Table 4-(b)) and the y-axis represents the ranking based on (raw value of) the composite measure.

Based on the values presented in Figure 5, we find the correlation between MCC and its neg/pos-normalized version to be quite strong ($r = 0.920$). On the other hand, the correlation of AUC of ROC, F1-value, G-mean and Balance with their respective neg/pos-normalized versions to be moderate (i.e. $r = 0.486$, $r = 0.469$, $r = 0.430$ and $r = 0.401$, respectively). Therefore, we recommend using MCC rather than the F1-value, AUC of ROC, G-mean and Balance.

Regarding the identification of successful/unsuccessful prediction, the composite measures are sometimes not useful. For example, data set no. 1 (MYLN) has high F1-value (0.728), AUC of ROC (0.857), MCC (0.513), G-mean (0.738) and Balance (0.707) as shown in Table 5, but according to the Table 3, its neg/pos-normalized recall, precision, specificity, NPV are not all positive, therefore the prediction for the data set is considered to be unsuccessful. On the other hand, data set no. 6 (CAML) has lower F1-value (0.345), AUC of ROC (0.669), MCC (0.252), G-mean (0.499) and Balance (0.480), but its neg/pos-normalized recall, precision, specificity, NPV are all positive, so the prediction for this data set is considered to be successful.

V. THREATS TO VALIDITY

We provide a discussion on the validity of the proposed method in terms of three commonly adopted experimental validation approaches, i.e. internal validity, external validity and construct validity.

Internal validity refers to the extent to which the observed effect is a consequence of the presumed cause. In our case,

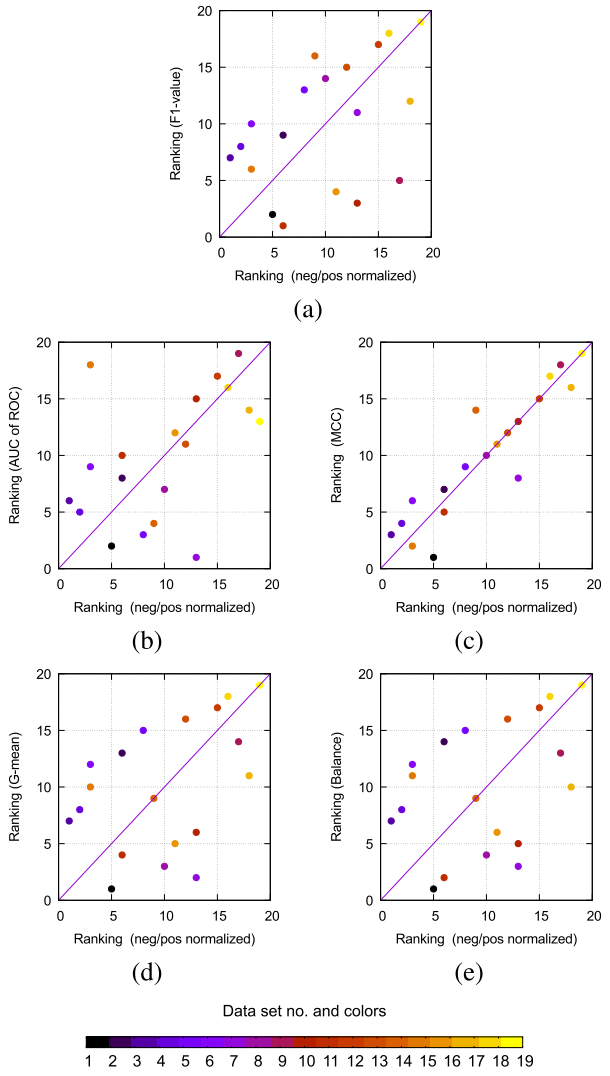

FIGURE 5. Relationship between the rankings based on (a) F1-value, (b) AUC of ROC, (c) MCC, (d) G-mean, (e) Balance and the ranking based on neg/pos normalized accuracy measures (see Table 4-(b)). Correlations coefficients are 0.469, 0.486, 0.920, 0.430 and 0.401, respectively.

one possible issues of internal validity are worth mentioning. This issue is that we used a single prediction method. In that respect, our important future work is to employ other prediction methods such as support vector machines and logistic regression models to increase the validity of the result.

External validity refers to the generalization of the results. In this study, we address external validity by using 38 releases of 19 open-source software project data sets with diverse characteristics obtained from different sources. Namely, they vary in number (i.e. the number of bugs), and project variables, as well as origin (i.e. recording organization) and recording period. In addition, we applied our method to four different accuracy measures Precision, Recall, NPV and Specificity. Our future work is to employ more data sets and accuracy measures to increase the generalization of the results.

Construct validity refers to the relevance and capability of the observations and measurements in evaluating the posed hypothesis. In this study, we compare the results of the proposed normalization with the (raw) values of conventional measures. It is our future work to employ other methods (e.g. human evaluation) to increase the validity of our work.

VI. CONCLUSION AND FUTURE PROSPECTS

This paper describes a way to compute *expected values* of accuracy measures based on all possible prediction outcomes. Based on the expected values, we define *neg/pos-normalized accuracy measures* as the difference between the actual and expected values divided by the standard deviation of all possible prediction outcomes. A case study of defect prediction with 19 data sets show that even a low accuracy value (e.g. `precision` < 0.1) could be considered to indicate a successful prediction (e.g. the case of data set no.14 (ECOS)), and a high accuracy value (e.g. `precision` > 0.8) can imply failure (e.g. the case of data set no.9 (LOG4) because it is too close to the baseline), depending on the composition (i.e. neg/pos ratio) of the data set. We also compare our ranking based on the neg/pos-normalized measures with conventional ranking using F1-value, AUC of ROC, MCC, G-mean and Balance, and found that MCC showed the highest correlation ($r = 0.920$) with our ranking; thus, we recommend using MCC rather than F1-value, AUC of ROC, G-mean and Balance.

As future work, we will employ other prediction models with more data sets and accuracy measures to increase the generalization of the results and to increase the construct validity of our work.

APPENDIX A TOY EXAMPLE FOR COMPUTATION OF EXPECTED VALUES OF ACCURACY MEASURES

Now let us study how this procedure works on a toy example. Consider the hypothetical test data with $T = 5$ modules shown in Table 6. As shown in the first row of this table, A^+ is 2 and A^- is 3.

The lower part of Table 6 lists such predictions (i.e. vectors π_i) for which $P^+ = 2$ and $P^- = 3$.[14] As seen in this table, the number of prediction results with this composition is 10,

$$\#\Big(\Pi(2,3)\Big) = \binom{5}{2} = 10.$$

In addition, the expected value of π, i.e $\mathbf{E}(\Pi)$ can be computed as the average of all π_is as

$$\mathbf{E}(\Pi) = \frac{\sum_{i=1}^{10} \pi_i}{10} = \big[0.4\ 0.4\ 0.4\ 0.4\ 0.4\big]. \tag{16}$$

Without loss of generality, let us focus on the 1st module, which is an actual positive. There are 10 predictions for this module, i.e. one in each π_i, $i \in [1, 10]$. In addition, 2/5 of

[14] In this table, we sorted the vectors π_i in decreasing order, as if they are binary numbers. Nevertheless, the order of sorting is not important for our problem.

TABLE 6. Example of test data and possible predictions.

		1st module	2nd module	3rd module	4th module	5th module
Actual condition		1	1	0	0	0
Set of possible permutations Π	π_1	1	1	0	0	0
	π_2	1	0	1	0	0
	π_3	1	0	0	1	0
	π_4	1	0	0	0	1
	π_5	0	1	1	0	0
	π_6	0	1	0	1	0
	π_7	0	1	0	0	1
	π_8	0	0	1	1	0
	π_9	0	0	1	0	1
	π_{10}	0	0	0	1	1

those 10 predictions are True Positive, i.e.

$$\frac{2}{5} \cdot 10 = 4.$$

Since there are 2 actual positives and there are 4 True Positives for each, the total number of True Positives concerning all the modules is

$$2 \cdot \frac{2}{5} \cdot 10 = 8.$$

Examining all predictions $\pi_1 \sim \pi_{10}$ in Table 6, one may see that the total number of True Positives concerning all 5 modules is indeed (i.e. $TP = 8$). Therefore, the expected TP is computed as *TP per prediction*,

$$\mathbf{E}(TP) = \frac{8}{10}.$$

Similarly, examining $\pi_1 \sim \pi_{10}$, one can see that there are a total of 12 FPs, 18 TNs and 12 FNs in Table 6. Thus, their expected values are $\mathbf{E}(FP) = 1.2$, $\mathbf{E}(TN) = 1.8$, $\mathbf{E}(FN) = 1.2$.

Based on these expected values, it is possible to compute the expected values of accuracy measures (e.g. $\mathbf{E}(\texttt{Precision})$, $\mathbf{E}(\texttt{Recall})$, $\mathbf{E}(\texttt{Npv})$, etc.). For example, concerning the example illustrated in Table 6,

$$\begin{aligned}\mathbf{E}(\texttt{Precision}) &= \frac{\mathbf{E}(TP)}{\mathbf{E}(TP) + \mathbf{E}(FP)} \\ &= \frac{0.8}{0.8 + 1.2} \\ &= 0.4.\end{aligned}$$

Relating to the same toy example, one can calculate the expected value of recall $\mathbf{E}(\texttt{Recall})$ as, once again, 0.4.

By using such expected values as baseline criteria, the quality of a particular prediction model can be evaluated. For instance, a model which yields as prediction result the 2nd permutation in Table 6) attains `precision` of 0.5 and a `recall` of 0.5, both of which are larger than the corresponding expected values. In that respect, this prediction model can be judged to be a good one.

REFERENCES

[1] G. Abaei, W. Z. Tah, J. Z. W. Toh, and E. S. J. Hor, "Improving software fault prediction in imbalanced datasets using the under-sampling approach," in *Proc. 11th Int. Conf. Softw. Comput. Appl.*, Feb. 2022, pp. 41–47.

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

[3] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proc. 6th Int. Conf. Predictive Models Softw. Eng. (PROMISE)*, 2010, pp. 1–10.

[4] M. Jureczko and D. Spinellis, "Using object-oriented design metrics to predict software defects," *Models Methods Syst. Dependability, Oficyna Wydawnicza Politechniki Wrocławskiej*, pp. 69–81, Jun. 2010.

[5] T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction," in *Proc. 22nd IEEE Int. Conf. Tools Artif. Intell.*, Oct. 2010, pp. 137–144.

[6] A. Kaur and I. Kaur, "Empirical evaluation of machine learning algorithms for fault prediction," *Lect. Notes Softw. Eng.*, vol. 2, no. 2, p. 176, 2014.

[7] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *Proc. Int. Conf. Mach. Learn.*, 1997, vol. 97, no. 1, pp. 179–186.

[8] Y. Kamei, S. Matsumoto, A. Monden, K.-I. Matsumoto, B. Adams, and A. E. Hassan, "Revisiting common bug prediction findings using effort-aware models," in *Proc. IEEE Int. Conf. Softw. Maintenance*, Sep. 2010, pp. 1–10.

[9] 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.

[10] J. C. Munson and T. M. Khoshgoftaar, "The detection of fault-prone programs," *IEEE Trans. Softw. Eng.*, vol. 18, no. 5, p. 423, 1992.

[11] 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.

[12] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognit.*, vol. 30, pp. 1145–1159, Jul. 1997.

[13] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[14] 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.

[15] R. Li and S. Wang, "An empirical study for software fault-proneness prediction with ensemble learning models on imbalanced data sets," *J. Softw.*, vol. 9, no. 3, pp. 697–704, Mar. 2014.

[16] V. Garcia, R. A. Mollineda, and J. S. Sanchez, "Theoretical analysis of a performance measure for imbalanced data," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 617–620.

[17] 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.

[18] S. Morasca and L. Lavazza, "On the assessment of software defect prediction models via ROC curves," *Empirical Softw. Eng.*, vol. 25, no. 5, pp. 3977–4019, Sep. 2020.
[19] 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.* Cham, Switzerland: Springer, 2006, pp. 1015–1021.
[20] A. Tharwat, "Classification assessment methods," *Appl. Comput. Informat.*, vol. 17, no. 1, pp. 168–192, Jul. 2020.
[21] 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.*, Apr. 2015, pp. 1644–1647.
[22] 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.
[23] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," 2020, *arXiv:2010.16061*.
[24] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 1–13, 2020.
[25] 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.
[26] L. A. Jeni, J. F. Cohn, and F. De La Torre, "Facing imbalanced data-recommendations for the use of performance metrics," in *Proc. Humaine Assoc. Conf. Affect. Comput. Intell. Interact.*, Sep. 2013, pp. 245–251.
[27] Y. Liu, J. Cheng, C. Yan, X. Wu, and F. Chen, "Research on the Matthews correlation coefficients metrics of personalized recommendation algorithm evaluation," *Int. J. Hybrid Inf. Technol.*, vol. 8, no. 1, pp. 163–172, Jan. 2015.
[28] Q. Zhu, "On the performance of Matthews correlation coefficient (MCC) for imbalanced dataset," *Pattern Recognit. Lett.*, vol. 136, pp. 71–80, Aug. 2020.
[29] A. Monden, J. Keung, S. Morisaki, Y. Kamei, and K.-I. Matsumoto, "A heuristic rule reduction approach to software fault-proneness prediction," in *Proc. Asia–Pacific Softw. Eng. Conf.*, vol. 1, 2012, pp. 838–847.
[30] T. Watanabe, A. Monden, Y. Kamei, and S. Morisaki, "Identifying recurring association rules in software defect prediction," in *Proc. IEEE/ACIS 15th Int. Conf. Comput. Inf. Sci. (ICIS)*, Jun. 2016, pp. 1–6.
[31] Y. Kamei. (2016). *Software Engineering Data Repository for Research and Education*. Accessed: Aug. 4, 2022. [Online]. Available: http://analytics.jpn.org/SEdata/
[32] K. E. Bennin, J. Keung, A. Monden, Y. Kamei, and N. Ubayashi, "Investigating the effects of balanced training and testing data sets on effort-aware fault prediction models," in *Proc. IEEE Comput. Softw. Appl. Conf.*, Jun. 2016, pp. 154–163.
[33] T. Menzies, R. Krishna, and D. Pryor. (2017). *The SeaCraft Repository of Empirical Software Engineering Data*. Accessed: Aug. 18, 2022. [Online]. Available: https://zenodo.org/communities/seacraft
[34] T. M. Khoshgoftaar, X. Yuan, and E. B. Allen, "Balancing misclassification rates in classification-tree models of software quality," *Empirical Softw. Eng.*, vol. 5, no. 4, pp. 313–330, 2000.
[35] S. Huda, K. Liu, M. Abdelrazek, A. Ibrahim, S. Alyahya, H. Al-Dossari, and S. Ahmad, "An ensemble oversampling model for class imbalance problem in software defect prediction," *IEEE Access*, vol. 6, pp. 24184–24195, 2018.
[36] R. Malhotra and S. Kamal, "An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data," *Neurocomputing*, vol. 343, pp. 120–140, May 2019.
[37] S.-J. Yen and Y.-S. Lee, "Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset," in *Intelligent Control and Automation*. Berlin, Germany: Springer, 2006, pp. 731–740.
[38] C. Liu, D. Yang, X. Xia, M. Yan, and X. Zhang, "A two-phase transfer learning model for cross-project defect prediction," *Inf. Softw. Technol.*, vol. 107, pp. 125–136, Mar. 2019.
[39] Z. Xu, S. Pang, T. Zhang, X.-P. Luo, J. Liu, Y.-T. Tang, X. Yu, and L. Xue, "Cross project defect prediction via balanced distribution adaptation based transfer learning," *J. Comput. Sci. Technol.*, vol. 34, no. 5, pp. 1039–1062, Sep. 2019.
[40] 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.*, 2004, pp. 417–428.
[41] N. Gayatri, S. Nickolas, A. Reddy, and R. Chitra, "Performance analysis of data mining algorithms for software quality prediction," in *Proc. Int. Conf. Adv. Recent Technol. Commun. Comput.*, 2009, pp. 393–395.
[42] A. Nadi and H. Moradi, "Increasing the views and reducing the depth in random forest," *Exp. Syst. Appl.*, vol. 138, Dec. 2019, Art. no. 112801.
[43] K. Shah, H. Patel, D. Sanghvi, and M. Shah, "A comparative analysis of logistic regression, random forest and KNN models for the text classification," *Augmented Hum. Res.*, vol. 5, no. 1, pp. 1–16, 2020.
[44] J. Keung, E. Kocaguneli, and T. Menzies, "Finding conclusion stability for selecting the best effort predictor in software effort estimation," *Automated Softw. Eng.*, vol. 20, no. 4, pp. 543–567, Dec. 2013.
[45] P. Phannachitta, J. Keung, A. Monden, and K. Matsumoto, "A stability assessment of solution adaptation techniques for analogy-based software effort estimation," *Empirical Softw. Eng.*, vol. 22, no. 1, pp. 474–504, Feb. 2017.
[46] R. Taylor, "Interpretation of the correlation coefficient: A basic review," *J. Diagnostic Med. Sonogr.*, vol. 6, no. 1, pp. 35–39, 1990.
[47] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[48] T. Fawcett, "ROC graphs: Notes and practical considerations for researchers," HP Labs, Palo Alto, CA, USA, Tech. Rep., HPL-2003-4, 2003.

MAOHUA GAN received the B.E. degree in software engineering from Northwestern Polytechnical University, in 2015, and the M.S. degree in information science from Okayama University, in 2020, where he is currently pursuing the Ph.D. degree with the Division of Industrial Innovation Sciences, Graduate School of Natural Science and Technology. His research interest includes software measurement and analytics.

ZEYNEP YÜCEL (Member, IEEE) received the B.S. degree in electrical engineering from Boğaziçi University, Istanbul, Turkey, and the M.S. and Ph.D. degrees in electrical engineering from Bilkent University, Ankara, Turkey, in 2005 and 2010, respectively. She was a Postdoctoral Researcher at ATR Laboratories, Kyoto, Japan, for five years, before being awarded a JSPS Fellowship, in 2016. She is currently an Associate Professor with Okayama University, Japan. Her research interests include robotics, signal processing, computer vision, and pattern recognition.

AKITO MONDEN (Member, IEEE) received the B.E. degree in electrical engineering from Nagoya University, in 1994, and the M.E. and D.E. degrees in information science from the Nara Institute of Science and Technology (NAIST), in 1996 and 1998, respectively. He is currently a Professor with the Graduate School of Natural Science and Technology, Okayama University, Japan. His research interests include software measurement and analytics, and software security and protection. He is a member of ACM, IEICE, IPSJ, and JSSST.