Empirical Software Engineering (2020) 25:3977–4019
https://doi.org/10.1007/s10664-020-09861-4

On the assessment of software defect prediction models via ROC curves

Sandro Morasca[1] · **Luigi Lavazza**[1]

Published online: 19 August 2020

Abstract

Software defect prediction models are classifiers often built by setting a threshold t on a defect proneness model, i.e., a scoring function. For instance, they classify a software module non-faulty if its defect proneness is below t and positive otherwise. Different values of t may lead to different defect prediction models, possibly with very different performance levels. Receiver Operating Characteristic (ROC) curves provide an overall assessment of a defect proneness model, by taking into account all possible values of t and thus all defect prediction models that can be built based on it. However, using a defect proneness model with a value of t is sensible only if the resulting defect prediction model has a performance that is at least as good as some minimal performance level that depends on practitioners' and researchers' goals and needs. We introduce a new approach and a new performance metric (the Ratio of Relevant Areas) for assessing a defect proneness model by taking into account only the parts of a ROC curve corresponding to values of t for which defect proneness models have higher performance than some reference value. We provide the practical motivations and theoretical underpinnings for our approach, by: 1) showing how it addresses the shortcomings of existing performance metrics like the Area Under the Curve and Gini's coefficient; 2) deriving reference values based on random defect prediction policies, in addition to deterministic ones; 3) showing how the approach works with several performance metrics (e.g., *Precision* and *Recall*) and their combinations; 4) studying misclassification costs and providing a general upper bound for the cost related to the use of any defect proneness model; 5) showing the relationships between misclassification costs and performance metrics. We also carried out a comprehensive empirical study on real-life data from the SEACRAFT repository, to show the differences between our metric and the existing ones and how more reliable and less misleading our metric can be.

Keywords Software defect prediction model · Software defect proneness · ROC · Thresholds · AUC · Gini

Communicated by: Martin Shepperd

✉ Sandro Morasca
sandro.morasca@uninsubria.it

Extended author information available on the last page of the article.

1 Introduction

Accurate estimation of which modules are faulty in a software system can be very useful to software practitioners and researchers. Practitioners can efficiently allocate scarce resources if they can predict which modules may need to undergo more extensive Verification and Validation than others. Researchers need to use quantitative, accurate module defect prediction techniques so they can assess and subsequently improve software development methods. In this paper, by the term "module," we denote any piece of software (e.g., routine, method, class, package, subsystem, system).

Several techniques have been proposed and applied in the literature for estimating whether a module is faulty (Beecham et al. 2010a; b; Hall et al. 2012; Malhotra 2015; Radjenović et al. 2013). We focus on those techniques that define defect prediction models (i.e., binary classifiers (Fawcett 2006)) by setting a threshold t on a defect proneness model (Huang et al. 2019), i.e., a scoring classifier that uses a set of independent variables. For instance, if the defect proneness model computes the probability that a module is faulty, a defect prediction model estimates a module faulty if its probability of being faulty is above or equal to t. The issue of defining the value of t has been addressed by several approaches in the literature (for instance, Alves et al. (2010), Erni and Lewerentz (1996), Morasca and Lavazza (2017), Schneidewind (2001), Shatnawi (2010), and Tosun and Bener (2009)).

The selection of t may greatly influence the estimates and the performance of the resulting defect prediction model. Thus, to evaluate a defect proneness model, one should evaluate the performance of the entire set of defect prediction models obtained with all possible values of t. Receiver Operating Characteristic (ROC) curves (see Section 4) have long been used to this end.

However, it is unlikely that all possible threshold values are used in practice. Suppose you have a defect proneness model for critical applications. It is unlikely that any sensible stakeholder selects a value of t corresponding to a high risk (e.g., to a 0.8 probability) of releasing a faulty module. Also, practitioners may not be able to confidently choose a "sharp" t value, corresponding to an exact probability value like 0.1. Instead, they may have enough information to know that the value of t should correspond to a risk level around 0.1, e.g., between 0.05 and 0.15. So, the evaluation should be restricted only to those defect prediction models that may be really used, depending on the goals and needs of individual practitioners.

In addition, some values of t are not useful because of two general reasons that do not depend on any specific practitioners' or researchers' goals and needs, hence they hold for every evaluation of defect prediction models.

First, suppose that a defect prediction model based on a set of independent variables and built with a given t does not perform better than a basic, reference defect prediction model that does not use any information from independent variables. Then, the defect prediction model should not be used, because one would be better off by using the simpler reference model. In other words, t is not an adequate threshold value for the defect proneness model.

Second, the prediction models obtained as t varies have different performance, but one should use in practice only those that perform well enough, based on some definition of performance and its minimum acceptable level.

So, in general, it may be ineffective and even misleading to evaluate the defect prediction models built with *all* possible values of t.

The goal of this paper is to propose an approach to assessing a given defect proneness model. We show how to use ROC curves and reference models to identify the defect prediction models that are worth using because they perform well enough for practical use

and outperform reference ones according to (1) standard performance metrics and (2) cost. Thus, we identify the values of t for which it is worthwhile to build and use defect prediction models. Our empirical validation shows the extent of the differences in the assessment of defect prediction models between our method and the traditional one.

Our approach helps practitioners compare defect prediction models and select those useful for their goals and needs. It allows researchers to assess software development techniques based only on those defect prediction models that should be used in practice, and not on irrelevant ones that may bias the results and lead to misleading conclusions.

Here are the main contributions of our proposal.

- We introduce a new performance metric, the Ratio of Relevant Areas (*RRA*). *RRA* can take into account only those parts of the ROC curve corresponding to thresholds for which it is worthwhile to build defect prediction models, i.e., the defect prediction models perform well enough according to some specified notion of performance. We also show how *RRA* can be customized to address the specific needs of different practitioners.
- We show that the Area Under the Curve (*AUC*) and Gini's coefficient (G) (Gini 1912) and other proposals are special cases of *RRA*, which, however, account for parts of the ROC curve corresponding to thresholds for which it is not worthwhile to build defect prediction models.
- We show how cost can be taken into account. We also provide an inequality that should be satisfied by all defect prediction models, regardless of the way they are built and of the specific misclassification costs.
- We show that choosing a performance metric (like *Precision*, *Recall*, etc.) for the assessment of defect prediction models is not simply a theoretical decision, but it equates to choosing a specific cost model.

We would like to clarify upfront that, in this paper, we are not interested in building the best performing models possible. Metrics like *AUC*, G, and *RRA* are used to assess existing, given models. We build models simply because we need them to demonstrate how our proposal works. To this end, we use 67 datasets from the SEACRAFT repository (https://zenodo.org/communities/seacraft). These datasets contain data about a number of measures and the faultiness of the modules belonging to real-life projects. At any rate, in the empirical validation illustrated in Section 10, we build defect proneness models that use all of the available measures so as to maximize the use of the available information about the modules' characteristics and possibly model performance as well.

The remainder of this paper is organized as follows. Section 2 recalls basic concepts of Software Defect Prediction along with the performance metrics that we use. Section 3 introduces the reference policies against which defect prediction models are evaluated. Sections 4 and 5 summarize the fundamental concepts underlying ROC curves and a few relevant issues. We show how to delimit the values of t that should be in general taken into account and we introduce *RRA* in Section 6. We show how *RRA* can be used based on several performance metrics in Section 7 and based on cost in Section 8. Section 9 compares *RRA* to *AUC* and G. We empirically demonstrate our approach on datasets from real-life applications in Section 10 and also highlight the insights that *RRA* can provide and traditional metrics can not. Section 11 illustrates the usefulness of our approach in Software Engineering practice. Threats to the validity of the empirical study are discussed in Section 12. Section 13 discusses related work. The conclusions and an outline for future research are in Section 14. Appendices A–F contain further details on some mathematical aspects of the paper.

Table 1 A confusion matrix

		Actual		
		Negative	Positive	
Estimated	Negative	TN (True Negatives)	FN (False Negatives)	EN = TN + FN (Estimated Negatives)
	Positive	FP (False Positives)	TP (True Positives)	EP = FP + TP (Estimated Positives)
		AN = TN + FP (Actual Negatives)	AP = FN + TP (Actual Positives)	n = AN + AP = EN + EP

2 Software Defect Prediction

The investigation of software defect prediction is carried out by first learning a defect prediction model (i.e., a binary classifier (Fawcett 2006)) on a set of data called the training set, and then evaluating its performance on a set of previously unseen data, called the test set. By borrowing from other disciplines, we use the labels "positive" for "faulty module" and "negative" for "non-faulty module". We denote by $\underline{z} = \langle z_1, z_2, \ldots \rangle$ the set of independent variables (i.e., features) used by a defect prediction model. Also, m will denote a module and we write 'm' for short, instead of writing "a module" or "a module m".

We use defect prediction models $fn(\underline{z},t)$ built by setting a threshold t on a defect proneness model $fp(\underline{z})$, so that $fn(\underline{z},t) = positive \Leftrightarrow t \leq fp(\underline{z})$ and $fn(\underline{z},t) = negative \Leftrightarrow fp(\underline{z}) < t$. For instance, we can use a Binary Logistic Regression (BLR) (Hardin and M 2002; Hosmer et al. 2013) model, which gives the probability that m is positive, to build a defect prediction model by setting a probability threshold t. Different values of t may lead to different classifiers, with different performance.

The performance of a defect prediction model can be assessed based on a confusion matrix, whose schema is shown in Table 1.

Table 2 shows the performance metrics we use in this paper. They include some of the most used ones and provide a comprehensive idea about the performance of a defect prediction model. The first three columns of Table 2 provide the name, the definition formula, and a concise explanation of the purpose of a metric. The other two columns are explained in Section 3.1.

Specifically, *Precision*, *Recall*, and *FM* (i.e., the F-measure or F-score (van Rijsbergen 1979)) assess performance with respect to the positive class, while Negative Predictive Value (*NPV*), *Specificity*, and a new metric that we call "Negative-F-measure" (*NM*), which mirrors *FM*, assess performance with respect to the negative class. Youden's J (Youden 1950) (also known as "Informedness"), *Markedness* (Powers 2012), and ϕ (also known as Matthews Correlation Coefficient (Matthews 1975)), which is the geometric mean of J and *Markedness*, are overall performance metrics.

Other metrics can be used as well. At any rate, using different metrics does not affect the way our approach works.

Table 2 Performance metrics for confusion matrices

Metric	Definition	Purpose	uni	Pop
Precision	$\frac{TP}{EP}$	proportion of estimated positives that are actual positives	$\frac{1}{1+k}$	$\frac{1}{1+k}$
Recall	$\frac{TP}{AP}$	proportion of actual positives that are estimated positive	p	$\frac{1}{1+k}$
FM	$\frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}$	overall evaluation for positives	$\frac{2p}{p(1+k)+1}$	$\frac{1}{1+k}$
NPV	$\frac{TN}{EN}$	proportion of estimated negatives that are actual negatives	$\frac{k}{1+k}$	$\frac{k}{1+k}$
Specificity	$\frac{TN}{AN}$	proportion of actual negatives that are estimated negative	$1-p$	$\frac{k}{1+k}$
NM	$\frac{2}{\frac{1}{NPV}+\frac{1}{Specificity}}$	overall evaluation for negatives	$\frac{2(1-p)k}{(1+k)(1-p)+k}$	$\frac{k}{1+k}$
J	$\frac{TP}{AP}-\frac{FP}{AN}$	overall evaluation for estimated positives	0	0
Markedness	$\frac{TP}{EP}-\frac{FN}{EN}$	overall evaluation for actual positives	0	0
ϕ	$\frac{TP\cdot TN-FP\cdot FN}{\sqrt{EN\cdot EP\cdot AN\cdot AP}}$	overall evaluation for Positives and negatives	0	0

3 Defining Reference Performance Values

The performance of the application of a defect prediction model on a dataset can be assessed by comparing the values obtained for one or more metrics against specified reference values, which set minimal performance standards. Classifiers not meeting these minimal standards should be discarded.

We use two methods to set reference values of performance metrics for defect prediction models: (1) methods based on random policies (see Section 3.1) and (2) deterministic ones (see Section 3.2). In both cases, our goal is to define reference values whose computation does not use any information about the individual modules.

3.1 Methods Based on Random Software Defect Prediction Policies

A random software defect prediction policy associates module m with a probability $p(m)$ that m is estimated positive. Thus, a random policy does not define a single defect prediction model, which deterministically labels each m. Rather, we have a set of defect prediction models, each of which is the result of "tossing a set of coins," each with probability $p(m)$ for each m.

The performance of a random policy can be evaluated based on the values of TP, TN, FP, and FN that can be expected, i.e., via an expected confusion matrix in which each cell

contains the expected value of the metric in the cell. These expected confusion matrices have already been introduced in Morasca (2014) for estimating the number of faults in a set of modules. The same metrics of Table 2 can be defined for random policies, by using the cells of the expected confusion matrix, e.g., *Precision* for a random policy is computed as $\frac{\mathbb{E}[TP]}{\mathbb{E}[EP]}$, where $\mathbb{E}[TP]$ and $\mathbb{E}[EP]$ are the expected values of TP and EP, respectively..

Our goal is to define random policies that use no information about m. Lacking any information about the specific characteristics of m, all modules must be treated alike. Thus, each m must be given the same probability p to be estimated positive, i.e., a uniform random policy must be used.

We use uniform random policies as references (see Sections 6 and 7) against which to evaluate the performance of defect prediction models. A classifier that performs worse than what can be expected of a uniform random policy should not be used. Our proposal for reference policies is along the lines of Langdon et al. (2016), Lavazza and Morasca (2017), and Shepperd and MacDonell (2012), where a random approach is used for defining reference effort estimation models. Also, completely randomized classifiers were used as reference models for defect prediction in the empirical studies of (Herbold et al. 2018) and in the application examples of Khoshgoftaar et al. (2001), though the performance of random classifiers was not taken into account in the definition of any performance metric. In what follows, we use "uni" to denote the uniform random policy for some specified value of p.

The expected confusion matrix for random policies depends on p. We use the value of p that sets the hardest challenge for models using variables, i.e., its Maximum Likelihood estimate $p = \frac{AP}{n}$. This special case of *uni*, which we call "pop" (as in "proportion of positives"), is one of the reference policies used in the empirical study of Section 10.

Table 3 shows the values of the cells of an expected confusion matrix for *uni* and *pop* policies.

In what follows, we write $k = \frac{AN}{AP}$ (following Flach 2003) to summarize properties of the underlying dataset. For instance, instead of $\frac{AP}{n}$, we write $\frac{1}{1+k}$.

Note that $\phi_{uni} = \phi_{pop} = 0$ (and $J_{uni} = J_{pop} = Markedness_{uni} = Markedness_{pop} = 0$), so *uni* and *pop* are never associated with the defectiveness of modules in any dataset. Thus, *uni*, and especially *pop*, can be used as reference policies in the evaluation of defect prediction models.

3.2 Deterministic Methods

We set a deterministic reference value for ϕ, which is the best known of the three overall performance metrics of Table 2, i.e., J, *Markedness*, and ϕ. The reference value of ϕ for the *uni* and *pop* policies is zero, which sets too low a standard against which to compare the ϕ

Table 3 Expected confusion matrix for *uni* and *pop* policies

		Actual					
		Negative		Positive		Est. Totals	
		uni	pop	uni	pop	uni	pop
Estimated	Negative	$(1-p)AN$	$\frac{AN^2}{n}$	$(1-p)AP$	$\frac{AP \cdot AN}{n}$	$(1-p)n$	AN
	Positive	$p \cdot AN$	$\frac{AP \cdot AN}{n}$	$p \cdot AP$	$\frac{AP^2}{n}$	$p \cdot n$	AP
	Act. Totals	AN		AP		n	

values of defect prediction models. Any model that provides a positive association between a defect prediction model and actual labels of modules, no matter how small, would be considered better than the standard. In our empirical study, we select $\phi = 0.4$ as a reference value for ϕ for medium/strong association, as it is halfway between the medium ($\phi = 0.3$) and strong ($\phi = 0.5$) values indicated in Cohen (1988).

We do not select deterministic reference values for the first six metrics in Table 2 because we can already set reference values based on the *pop* policy.

4 ROC: Basic Definitions and Properties

A Receiver Operating Characteristic (ROC) curve (Fawcett 2006) is a graphical plot that illustrates the diagnostic ability of a binary classifier $fn(\underline{z},t)$ with a scoring function $fp(\underline{z})$ as its discrimination threshold t varies.

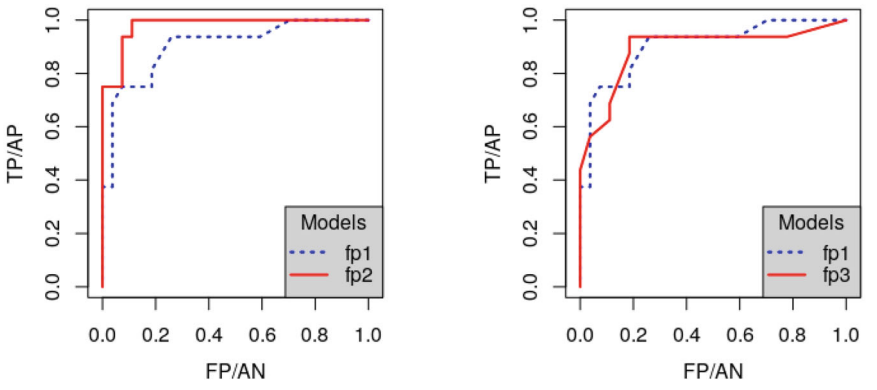
A ROC curve—which we denote as a function $ROC(x)$ —plots the values of $y = Recall = \frac{TP}{AP}$ against the values of $x = Fall\text{-}out = \frac{FP}{AN} = 1\text{–}Specificity$ computed on a test set for all the defect prediction models $fn(\underline{z},t)$ obtained by using all possible threshold values t. Examples of ROC curves are in Fig. 1.

The $[0, 1] \times [0, 1]$ square to which a ROC curve belongs is called the ROC space (Fawcett 2006). Given a dataset, each point (x, y) of the ROC space corresponds to a defect prediction model's confusion matrix, since the values of x and y allow the direct computation of TP and FP and the indirect computation of TN and FN, since AP and AN are known.

The two variables x and y are related to t in a (non-strictly) monotonically decreasing way. Hence, a ROC curve $ROC(x)$ is a non-strictly monotonically increasing function of x.

We now introduce the definition of Upper-left Rectangle of a point (x, y) of the ROC space, which will be used in the remainder of the paper.

Definition 1 *Upper-left Rectangle (ULR) of a Point.*
The Upper-left Rectangle $ULR(x,y)$ of a Point (x,y) is the closed rectangle composed of those points (x',y') of the ROC space such that $x' \leq x \wedge y' \geq y$.



(a) ROC curves of $\mathtt{fp}_1(\underline{z})$ and $\mathtt{fp}_2(\underline{z})$ (b) ROC curves of $\mathtt{fp}_1(\underline{z})$ and $\mathtt{fp}_3(\underline{z})$

Fig. 1 ROC curves of defect proneness models from the `berek` dataset

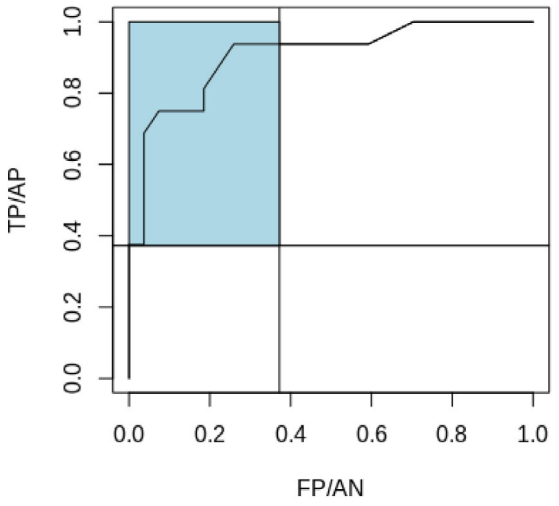


Fig. 2 ROC curve with straight lines $y = Recall_{pop} = \frac{1}{1+k}$ and $x = Fall{-}out_{pop} = \frac{1}{1+k}$. The highlighted rectangle is $ULR(\frac{1}{1+k}, \frac{1}{1+k})$

An example of ULR is represented by the highlighted rectangle in Fig. 2, which shows $ULR(\frac{1}{1+k}, \frac{1}{1+k})$.

All of the points of $ULR(x,y)$ are not worse than (x,y) itself for any sensible performance metric. They correspond to defect prediction models with no more false negatives nor more false positives than the defect prediction model corresponding to (x,y). Point (0, 1) has no other point in its $ULR(0,1)$, so it corresponds to the best classifier, which provides perfect estimation.

ROC curves have long been used in Software Defect Prediction (Arisholm et al. 2007; Beecham et al. 2010b; Catal 2012; Catal and Diri 2009; Singh et al. 2010), typically to have an overall evaluation of the performance of the defect prediction models learned based on $\underline{z}$ with all possible values of t.

The evaluation of $fn(\underline{z},t)$ for all values of t, i.e., the overall evaluation of $fp(\underline{z})$, is typically carried out by computing the Area Under the Curve.

Definition 2 *Area Under the Curve (AUC).*

The Area Under the Curve is the area below $ROC(x)$ in the ROC space.

Table 4 Evaluation of AUC and G

AUC range	G range	Evaluation
$AUC = 0.5$	$G = 0$	Totally random, as good as tossing a coin
$0.5 < AUC < 0.7$	$0 < G < 0.4$	Poor, not much better than a coin toss
$0.7 \leq AUC < 0.8$	$0.4 \leq G < 0.6$	Acceptable
$0.8 \leq AUC < 0.9$	$0.6 \leq G < 0.8$	Excellent
$0.9 \leq AUC$	$0.8 \leq G$	Outstanding

The longer $ROC(x)$ lingers close to the left and top sides of the ROC space, the larger AUC. Since the total area of the ROC space is 1, the closer AUC is to 1, the better. Hosmer et al. (2013) propose the intervals in Table 4 as interpretation guidelines for AUC as a measure of how well $fn(\underline{z},t)$ discriminates between positives and negatives for all values of t.

When comparing defect proneness models, $fp_1(\underline{z}_1)$ (associated with AUC_1) is preferred to $fp_2(\underline{z}_2)$ (associated with AUC_2) if and only if $AUC_1 > AUC_2$ (Hanley and McNeil 1982).

The Gini coefficient $G = 2\ AUC–1$ is a related metric also used for the same purposes (Gini 1912). G takes values in the [0, 1] range and was defined in such a way as to be applied only to ROC curves that are never below the diagonal $y = x$. As there is a one-to-one functional correspondence between them, AUC and G provide the same information. Column "G range" in Table 4 shows how Hosmer at al.'s guidelines for AUC (Hosmer et al. 2013) can be rephrased in terms of G.

Other metrics have been defined in addition to AUC and G. We concisely review some of them in Section 13.1.

5 Evaluation Issues

ROC curves have been widely studied and used in several fields, and a few issues have been pointed out about their definition and evaluation. We concisely review them in Section 13. Here, we focus on two issues about the use of AUC as a sensible way for providing (1) an evaluation of a defect proneness model and (2) a comparison between two defect proneness models.

5.1 Evaluation of a Defect Proneness Model: The Diagonal

The diagonal of the ROC space represents the expected performance of random policies. Table 3 shows that, given a value of p, $\mathbb{E}[FP] = p \cdot AN$ and $\mathbb{E}[TP] = p \cdot AP$, so $\mathbb{E}[x] = \frac{\mathbb{E}[FP]}{AN} = p$ and $\mathbb{E}[y] = \frac{\mathbb{E}[TP]}{AP} = p$. Thus, $\mathbb{E}[x] = \mathbb{E}[y] = p$, i.e., the diagonal $y = x$ is the expected ROC curve under a random policy, for each possible value of p. Since the points in $ULR(x, y)$ are not worse than (x, y), the upper-left triangle of the ROC space delimited by the diagonal is also the set of points corresponding to defect prediction models whose performance is not worse than the expected performance of random policies. In practice, the upper-left triangle is the truly interesting part of the ROC space when building useful defect prediction models. It is well-known (Fawcett 2006) that, if a classifier corresponds to a point in the lower-right triangle, a better classifier can be obtained simply by inverting its estimates.

However, AUC is computed by also taking into account the lower-right triangle of the ROC space. Notice that AUC is the area between $ROC(x)$ and the reference ROC curve $y = 0$, which corresponds to a defect prediction model that estimates $TP = 0$ for all values of t. In other words, AUC quantifies how different a ROC curve is from this extremely badly performing classifier—even worse than what is expected of any random policy. In practice, instead, AUC is to be compared to random policies, as Table 4 shows.

Using random policies, characterized by $y = x$, as the reference instead of $y = 0$ appears more adequate. Instead of the area under the curve in the entire ROC space, one can use the area under the curve in the upper-left triangle, and normalize it by the area of the triangle, i.e., $1/2$. This is actually the value of the Gini coefficient G, when $ROC(x)$ is entirely

above the diagonal. If that is not the case, one can define a modified ROC curve *ROC'*(x) that coincides with *ROC*(x) when *ROC*(x) is above the diagonal, and coincides with the diagonal otherwise. Practically, this corresponds to using the defect prediction models for all of those values of t in which one obtains better performance than random policies, and random policies otherwise, which are an inexpensive backup estimation technique one can always fall back on.

However, even this modified version of G may not be satisfactory for practitioners' and researchers' goals, which may require that one or more performance metrics of a defect prediction model be higher than some specified minimum reference values, and not simply better than a random classifier. The approach proposed in this paper (see Section 6) extends the idea of comparing the performance of models with respect to random policies by taking into account specific performance metrics, and compares their values when they are computed with the defect prediction models obtained based on a defect proneness model and all possible thresholds.

5.2 Comparison of Defect Proneness Models with *AUC*: Deceiving Cases

Suppose we have two defect proneness models $fp_1(\underline{z})$ and $fp_2(\underline{z})$. Figure 1a shows that the ROC curve of $fp_2(\underline{z})$ is always above the ROC curve of $fp_1(\underline{z})$, i.e., the *Recall* and *Fall-out* of $fp_2(\underline{z})$ are never worse than those of $fp_1(\underline{z})$. Accordingly, the *AUC* of $fp_2(\underline{z})$ is greater than the *AUC* of $fp_1(\underline{z})$. $fp_2(\underline{z})$ is at least as good as $fp_1(\underline{z})$ for all choices of t, but we do not need *AUC* to decide which defect proneness model is better.

Figure 1b instead shows two intersecting ROC curves. It is not straightforward to decide which defect proneness model is better by simply looking at the ROC curves, since neither curve dominates the other. Using *AUC* would have us conclude that $fp_1(\underline{z})$ is not worse than $fp_3(\underline{z})$, since the *AUC* of $fp_1(\underline{z})$is greater than the *AUC* of $fp_3(\underline{z})$. However, Fig. 1b also shows that the *AUC* of $fp_1(\underline{z})$ is greater than that of $fp_3(\underline{z})$ mainly because the ROC curve of $fp_1(\underline{z})$ is above the ROC curve of $fp_3(\underline{z})$ when *FP/AN*$>$ 0.6, i.e., when *Fall-out* is quite high and the defect prediction models obtained with *both* defect proneness models provide quite bad estimates.

Thus, just because the *AUC* of a defect proneness model is greater than that of another does not automatically mean that the former defect proneness model is preferable. Instead, we should restrict the comparison to the zone (i.e., the threshold range) where the defect prediction models obtained behave "acceptably".

In the following sections, we propose methods to "purge" *AUC* from the noise originated by defect prediction models that do not perform well enough. The resulting indications are expected to be more reliable, hence more useful in practice and also more correct from a theoretical point of view.

6 Evaluation Based on Relevant Areas

Suppose we select a performance metric *PFM* and a random or deterministic method *MTD* for selecting a reference value to evaluate the acceptability of a defect prediction model. Let us denote by PFM_{MTD} the reference performance value of *MTD* when evaluated via *PFM*. For instance, if we take *FM* as *PFM* and *pop* as *MTD*, we have $FM_{pop} = \frac{1}{1+k}$, as shown in Table 2.

Among the defect prediction models that can be generated with $fn(\underline{z},t)$, the ones that should be considered performing sufficiently well are those that provide a better value of

PFM than the reference value, e.g., better than what can be expected of a random policy. These are the practically useful defect prediction models, hence the ones that should be taken into account when evaluating the overall performance of $fp(\underline{z})$. For instance, if we decide to assess the performance of $fn(\underline{z},t)$ based on $PFM_1 = Recall$ and $MTD_1 = pop$, we should only take into account those values of t in which $fn(\underline{z},t)$ has $Recall > Recall_{pop} = \frac{1}{1+k}$.

Note that *there are always values of t that are so small or so large to make estimates' performance according to some PFM similar or even worse than the performance with a reference policy*. Hence, it does not make sense to evaluate $fn(\underline{z},t)$ for all the values of t. Instead, we take into account the points (x, y) of a ROC curve that satisfy inequality $y > Recall_{pop} = \frac{1}{1+k}$ (see Table 2), i.e., the points above the horizontal straight line $y = \frac{1}{1+k}$.

Recall captures one aspect of performance, mainly based on true positives, but other aspects can be of interest. Suppose we decide to assess the performance of $fn(\underline{z},t)$ based on $PFM_2 = Fall\text{-}out$, which captures performance by taking into account the false positives, and $MTD_2 = pop$. We should only take into account the values of t in which $fn(\underline{z},t)$ has $Fall\text{-}out < Fall\text{-}out_{pop} = 1 - Specificity_{pop} = \frac{1}{1+k}$. We are thus interested only in the points satisfying inequality $x < \frac{1}{1+k}$, i.e., left of the vertical straight line $x = \frac{1}{1+k}$.

If we are interested in the points of the ROC curve that are better than *pop* for both *Recall* and *Fall-out*, then both inequalities must be satisfied, and the evaluation must consider only the points of the ROC curve in $ULR(\frac{1}{1+k}, \frac{1}{1+k})$, i.e., the highlighted rectangle in Fig. 2.

It is up to the practitioners and researchers to decide which metrics are of interest for their goals. For instance, they can use *FM* as PFM_1 and *NM* as PFM_2 and *pop* as both MTD_1 and MTD_2, to consider defect prediction models that perform better than *pop* for both *FM* and *NM*. The points of the ROC curve to take into account are represented in Fig. 3a, above and to the left of the two oblique straight lines with equations $y = \frac{k}{2k+1}x + \frac{1}{2k+1}$ for *FM* and $y = (k+2)x - 1$ for *NM*, as we show in Table 5.

More generally, other metrics and reference policies may be defined and used well beyond the ones illustrated in this paper. Different choices of metrics and reference policies may lead to delimiting any subset of the ROC space. Clearly, if one is interested in using several metrics and several corresponding reference policies, the subset of the ROC space is the intersection of the single subsets, each of which is built by means of a metric and a reference policy.

However, not all ROC space subsets are useful or sensible. We introduce the notion of "Region of Interest," to define which ones should be used.

Table 5 Borders for performance metrics for random policies

Metric	Formula	Border	uni	pop
Precision	$\frac{y}{kx+y}$	$y = \frac{c}{1-c}kx$	$y = x$	$y = x$
Recall	y	$y = c$	$y = p$	$y = \frac{1}{1+k}$
F-measure	$\frac{2y}{kx+y+1}$	$y = \frac{c}{2-c}kx + \frac{c}{2-c}$	$y = \frac{p}{pk+1}kx + \frac{p}{pk+1}$	$y = \frac{k}{2k+1}x + \frac{1}{2k+1}$
NPV	$\frac{k(1-x)}{k(1-x)+1-y}$	$y = \frac{1-c}{c}k(x-1)+1$	$y = x$	$y = x$
Specificity	$1 - x$	$x = 1 - c$	$x = p$	$x = \frac{1}{1+k}$
NM	$\frac{2k(1-x)}{k(1-x)+(1-y)+k}$	$y = \frac{2-c}{c}kx - 2\frac{1-c}{c}k + 1$	$y = \frac{1-p+k}{1-p}x - \frac{p}{1-p}k$	$y = (k+2)x - 1$

Definition 3 *Region of Interest (RoI).*
A subset of the ROC space is said to be a Region of Interest (*RoI*) if and only if it contains the upper-left rectangles of all of its points, i.e.,

$$\forall\, x, y(x, y) \in RoI \Rightarrow ULR(x, y) \subseteq RoI \tag{1}$$

The border of a RoI is the part of the boundary of the RoI in the interior of the ROC space, i.e., it is the boundary of the RoI without its parts that also belong to the ROC space perimeter. The border of the *RoI* is the part of its boundary that really provides information on how the *RoI* is delimited, since the perimeter of the ROC space can be taken for granted as a delimitation.

The union of the light blue and grey regions in Fig. 3a is an example of a *RoI*, in which $FM \geq FM_{pop} \wedge NM \geq NM_{pop}$ (see Section 7.1). An example of a subset of the ROC space that is not a *RoI* is in Fig. 3b.

RoIs have a few properties, which we prove in Appendix A.

- The intersection of any number of *RoIs* is a *RoI*.
- The intersection of any number of *RoIs* is nonempty.
- A *RoI* is connected.
- The smallest *RoI* to which a point (x, y) belongs is $ULR(x, y)$.
- A *RoI* always contains point (0, 1).
- The border of a *RoI* is a (non necessarily monotonically) increasing function. Thus, graphically, a *RoI* is above and to the left of its border.

With a reference value derived from a random policy, the points on the border of a *RoI* correspond to unacceptable defect prediction models, as their performance is as good as what can be expected of a random policy. If, instead, the reference value is selected deterministically, all points on the boundary correspond to acceptable classifiers, e.g., one takes $\phi = 0.4$ as minimum ϕ value for an acceptable defect prediction model.

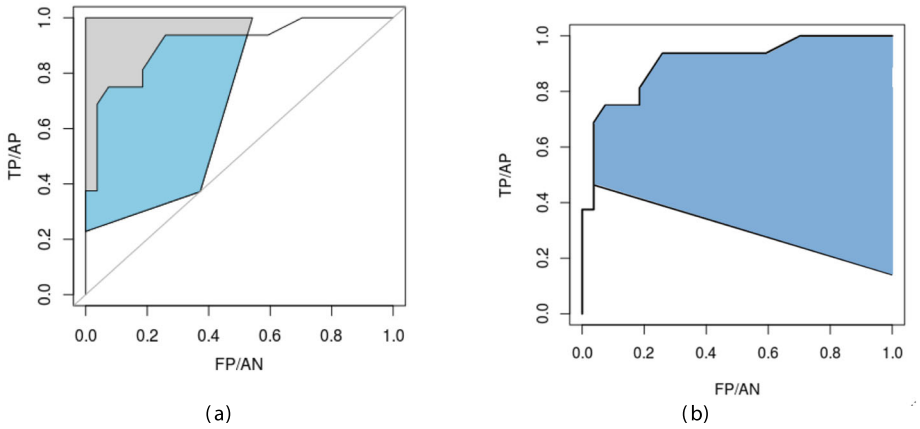


Fig. 3 A ROC and the *RoI* in which $FM \geq FM_{pop} \wedge NM \geq NM_{pop}$ (**a**) and a subset of the ROC space that is not a *RoI* (**b**)

In what follows, we implicitly assume that the points on the border of a *RoI* are included or not in the *RoI* depending on whether a reference value has been selected via a random policy or deterministically.

6.1 The Ratio of Relevant Areas

We propose to assess a fault-proneness model $fp(\underline{z})$ via the Ratio of the Relevant Areas (*RRA*), which takes into account only the *RoI* selected by a practitioner or researcher.

Definition 4 *Ratio of the Relevant Areas (RRA).*
The Ratio of the Relevant Areas of a ROC curve $ROC(x)$ in a *RoI* is the ratio of the area of the *RoI* that is below $ROC(x)$ to the total area of the *RoI*.

In Fig. 3a, the *RoI* is the union of the light blue and grey regions, in which the light blue region is the part of the *RoI* below $ROC(x)$. *RRA* is the ratio of the area of the light blue region to the area of the *RoI*.

AUC and *G* are special cases of *RRA*, obtained, respectively, when the *RoI* is the whole ROC space and the upper-left triangle. From a conceptual point of view, it is sounder to consider the area under the portion of the ROC curve in the *RoI* than to consider the areas under the entire ROC curve taken into account by *AUC* and *G*: *RoI* represents the part of the ROC space in which defect prediction models perform sufficiently well to be used.

Take, for instance, the case in which we use reference random policies of interest along with a set of performance metrics of interest to build a *RoI*. By considering the parts of $ROC(x)$ outside the *RoI*, one would also take into account values of t that make $fn(\underline{z}, t)$ worse than a random estimation method. When we know that a given defect prediction model is worse than a random policy for a set of performance metrics, it is hardly interesting to know precisely how well it performs. However, this is what *AUC* and *G* do.

Note that Definition 4 is quite general, as it allows the use of different reference policies for different performance metrics even when they are used together. For instance, one may be interested in the points of a ROC curve that are better than FM_{pop} and, at the same time, better than the *NM* value obtained with *uni* with $p = 0.7$. In what follows, however, we assume that the same reference policy is used for all of the performance metrics selected.

7 *RoIs* for Specific Performance Metrics and Reference Values

The requirement that a defect prediction model satisfy a minimum acceptable level c for a performance metric *PFM* corresponds to a *RoI* in the ROC space. We here show the equations of the borders of the *RoIs* corresponding to the metrics in Table 2. Appendix B shows how the equations for these borders were obtained, by explaining how these metrics, defined in terms of the cells of the confusion matrix, can be expressed in terms of x and y. These borders are akin to the "iso-performance lines" or "isometrics" proposed in Flach (2003), Provost and Fawcett (2001), and Vilalta and Oblinger (2000).

7.1 *RoIs* for Performance Metrics with Respect to the Positive and Negative Classes

Table 5 summarizes the formulas about the borders of the *RoIs* for the performance metrics with respect to the positive and the negative classes of Table 2 with the *uni* and *pop* reference

policies, for completeness. At any rate, we only use *pop* in the examples and in the empirical study of Section 10.

- Column "Formula" provides the definition of the performance metric in each row in terms of x and y. For instance, $Precision = \frac{y}{kx+y}$.
- Column "Border" shows the equation of the general straight line that represents the border of the *RoI* obtained when the performance metric corresponding to the row is given a constant value c. For instance, line $y = \frac{c}{1-c}kx$ includes the points where $Precision{=}c$. For practical usage, when we select a specific metric *PFM* and a reference method *MTD*, we replace the generic parameter c by the specific PFM_{MTD} chosen.
- Column "uni" shows the equation of the border when c is replaced by PFM_{uni} with probability p, where *PFM* is the metric in the corresponding row. This is the border of the *RoI* where defect prediction models have greater value of *PFM* than expected of the *uni* policy with probability p.
- Likewise, column "pop" shows the equation of the border when c is replaced by PFM_{pop}, where *PFM* is the metric in the corresponding row.

It can be shown that each equation in the "uni" column in Table 5 describes a pencil of straight lines (Cremona 2005) through point (p, p), i.e., $(\frac{1}{1+k}, \frac{1}{1+k})$ with *pop*.

Figure 4 shows the ROC curve already shown in Fig. 2, along with all the lines corresponding to the borders mentioned in Table 5 for *pop*. The portion of ROC above and to the left of the borders of all performance metrics ($ULR(\frac{1}{1+k}, \frac{1}{1+k})$, in this case) is quite small, compared to the entire ROC curve. Thus, there is a relatively small range where t provides $fn(\underline{z}, t)$ defect prediction models that perform better than *pop*, according to multiple metrics.

The borders in Table 5 follow expected change patterns when k, c, and p change. Higher values of c are associated with stricter constraints, e.g., the slope of the *Precision* straight

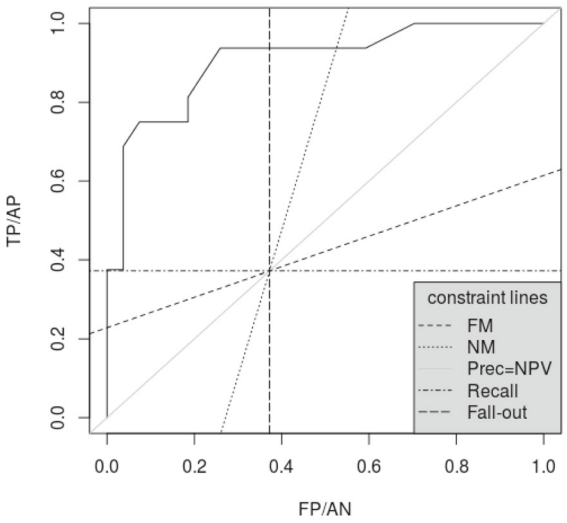


Fig. 4 ROC curve of *fp(WMC)* for the `berek` dataset with multiple *pop* constraints

Table 6 Borders for overall performance metrics

Metric	Formula	Border
J	$y - x$	$y = x + c$
Markedness	$\frac{k(y-x)}{(y+kx)(k(1-x)+(1-y))}$	$ck^2x^2 + 2ckxy + cy^2 +$ $-(ck^2 + ck + k)x - (ck - k + c)y = 0$
ϕ	$\frac{\sqrt{k}(y-x)}{\sqrt{(y+kx)(k(1-x)+(1-y))}}$	$(k + c^2k^2)x^2 + 2(c^2k - k)xy + (c^2 + k)y^2 +$ $-c^2(k^2 + k)x - c^2(k + 1)y = 0$

line $y = \frac{c}{1-c}kx$ increases with c. Appendix C details how these borders behave for each metric when k, c, and p change.

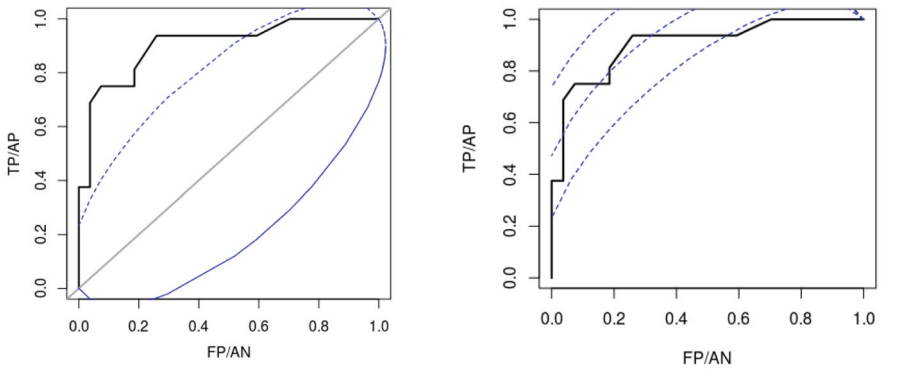
We use the *pop* policy in the empirical validation of Section 10. For notational convenience, we denote by $RoI(PFM_1, PFM_2, \ldots)$ the *RoI* defined by constraint $PFM_1 > PFM_{1,pop} \wedge PFM_2 > PFM_{2,pop} \wedge \ldots$), and by $RRA(PFM_1, PFM_2, \ldots)$ the value of *RRA* for $RoI(PFM_1, PFM_2, \ldots)$. For instance, $RRA(FM, NM)$ denotes the value of *RRA* for $RoI(FM, NM)$, i.e., the *RoI* with $FM > FM_{pop} \wedge NM > NM_{pop}$, e.g., the union of the light blue and grey regions in Fig. 3a.

7.2 *RoIs* for Overall Metrics

Table 6 shows the formula and the border obtained for each of the three overall performance metrics *PFM* in Table 2 when one sets a minimum acceptable value c for it, i.e., one requires $PFM \geq c$. Unlike Tables 5, and 6 does not contain columns "uni" and "pop," because we showed in Table 2 that J, *Markedness*, and ϕ are all equal to 0 under random policies. Therefore, a *RoI* is defined by means of a deterministically chosen value of c.

The lines for constant Youden's J are straight lines parallel to the diagonal. As for *Markedness*, it can be shown that the constant lines are parabolas, with symmetry axis $y = -kx + \frac{1+k}{2} + \frac{k(k-1)}{2c(1+k^2)}$. The details are in Appendix D.

ϕ has received the most attention among these three metrics in the past. It can be shown that the border for $\phi = c$ is an ellipse that goes through points (0, 0) and (1, 1) for all values



(a) Ellipse for φ = 0 .4. (b) Multiple constraints on φ determining RoIs.

Fig. 5 ROC curve of defect proneness vs. WMC $fp(WMC)$for `berek` dataset with constraints on ϕ

of c and k and intercepts the perimeter of the ROC space in four points (except for very special values of c and k). Appendix D shows the details of the analytic results we obtained.

Figure 5a shows the ellipse for the `berek` dataset (from the SEACRAFT repository) with $c = 0.4$, a value that represents medium/strong association between a defect prediction model and actual faultiness (Cohen 1988). Note that there are two unconnected parts of the ROC space in which $\phi \geq c$, delimited by the dashed part of the ellipse and the solid part of the ellipse. Based on Definition 3, only the upper-left part above the dotted arc of the ellipse is a legitimate *RoI*.

Figure 5b shows the borders of the *RoIs* associated with ϕ=0.4 (the lowest line), 0.6, and 0.8 (the highest line). By comparing Figs. 5 and 4, it is easy to see that the points of the ROC curve that satisfy the constraints mentioned in Table 5 also satisfy constraint $\phi \geq 0.4$. However, only a few points of the ROC curve (corresponding to a few selected values of t) satisfy constraint $\phi \geq 0.6$. No point of the ROC curve satisfies constraint $\phi \geq 0.8$.

8 Taking Cost into Account

We have so far considered the evaluation of defect prediction models with respect to the performance of estimates. Though the notion of performance is important, practitioners are usually also interested in other characteristics of estimates, such as the cost of misclassifying a faulty module as not faulty, or *vice versa*. As we show in Section 8.1, there is a clear relationship between the choice of a performance metric and the cost of misclassification.

We first show how to derive the border of a *RoI* based on the misclassification cost. Like most of the literature, we assume that each false negative (resp., positive) has the same cost c_{FN} (resp., c_{FP}), so total cost TC is Hand (2009)

$$TC = c_{FN} FN + c_{FP} FP \tag{2}$$

TC can be computed in terms of x and y of the ROC space as follows

$$TC = c_{FN} AP(1-y) + c_{FP} AN \cdot x = n\left(c_{FN}\frac{AP}{n}(1-y) + c_{FP}\frac{AN}{n}x\right) \tag{3}$$

By setting $\lambda = \frac{c_{FN}}{c_{FN}+c_{FP}}$ and dividing TC by $n(c_{FN} + c_{FP})$ (which is independent of the defect prediction model used), we can focus on Normalized Cost $NC = \frac{TC}{n(c_{FN}+c_{FP})}$ (Khoshgoftaar and Allen 1998)

$$NC = \lambda\frac{AP}{n}(1-y) + (1-\lambda)\frac{AN}{n}x = \lambda\frac{1}{1+k}(1-y) + (1-\lambda)\frac{k}{1+k}x \tag{4}$$

NC is related to Unitary Cost $UC = \frac{TC}{n} = (c_{FN} + c_{FP})NC$, so constraints on UC get immediately translated into constraints on NC and *vice versa*.

Usually, c_{FN} is much greater than c_{FP}, as false negatives have more serious consequences than false positives, financially and otherwise. Accordingly, λ is usually much closer to 1 than to $1/2$ (value $1/2$ corresponds to $c_{FN} = c_{FP}$).

8.1 Borders Based on Random Policies

For any random policy, thanks to basic properties of expected values, we have

$$\mathbb{E}[NC] = \lambda\frac{AP}{n}\mathbb{E}[(1-y)] + (1-\lambda)\frac{AN}{n}\mathbb{E}[x] \tag{5}$$

and for the *uni* and *pop* policies we have, based on Table 3

$$\mathbb{E}[NC_{uni}] = \lambda \frac{AP}{n} - p\left(\lambda - \frac{AN}{n}\right) \tag{6}$$

$$\mathbb{E}[NC_{pop}] = \frac{AP \cdot AN}{n^2} = \frac{k}{(1+k)^2} \tag{7}$$

Thus, one should use only those defect prediction models whose *NC* is less than the expected normalized cost of a random policy, i.e., $NC < \frac{AP \cdot AN}{n^2}$.

Since $\mathbb{E}[NC_{pop}]$ is independent of the specific cost per false negative or positive, Formula (7) provides a general result that applies to all defect prediction models, and regardless of the way they have been built, e.g., with or without using techniques based on defect proneness models and thresholds.

The value of $\mathbb{E}[NC_{pop}] = \frac{AP \cdot AN}{n^2} = \frac{AP}{n} - \frac{AP^2}{n^2}$ depends on the intrinsic characteristics of the dataset. As a function of AP, it describes a parabola with minimum ($\mathbb{E}[NC_{pop}] = 0$) when $AP = 0$ or $AP = n$ and maximum ($\mathbb{E}[NC_{pop}] = \frac{1}{4}$) when $AP = \frac{n}{2}$. Thus, $NC < \frac{AP \cdot AN}{n^2}$ is easier to satisfy when $AP \approx AN$, i.e., for balanced datasets, and more difficult when this is not the case. As we show in Section 8.2, lower values of *NC* call for better performing defect prediction models.

Via mathematical transformations, it can be shown that the borders of the *RoIs* that satisfy inequality $NC < \frac{k}{(1+k)^2}$ for different values of λ are represented by the following pencil of straight lines, with center in $(\frac{1}{1+k}, \frac{1}{1+k})$

$$y = \frac{1-\lambda}{\lambda} kx + 1 - \frac{k}{\lambda(1+k)} \tag{8}$$

It can be shown that the slope of the border decreases as λ varies from 0 to 1. Thus, the border rotates around center point $(\frac{1}{1+k}, \frac{1}{1+k})$ in a clockwise fashion from vertical straight line $x = \frac{1}{1+k}$ to horizontal straight line $y = \frac{1}{1+k}$. A special case occurs when $\lambda = \frac{k}{1+k}$, since the line is the diagonal $y = x$.

Recall that all of the straight lines for all of the performance metrics in Table 5 go through center point $(\frac{1}{1+k}, \frac{1}{1+k})$ too. Thus, they are special cases of the straight lines described in Formula (8), for specific values of λ. Specifically, we have, in increasing order of the value of λ: for *Fall-out*, $\lambda = 0$; for *NM*, $\lambda = \frac{k}{2(1+k)} = \frac{AN}{2n}$; for *Precision* and for *NPV*, $\lambda = \frac{k}{1+k} = \frac{AN}{n}$; for *FM*, $\lambda = \frac{2k+1}{2k+2} = \frac{1}{2} + \frac{AN}{2n}$; and for *Recall*, $\lambda = 1$.

Thus, the selection of any of these metrics is not simply an abstract choice on how to assess the performance of defect prediction models, but implies the choice of a specific cost model with a specific $\lambda = \frac{c_{FN}}{c_{FN}+c_{FP}}$, which implies a specific ratio $\frac{c_{FN}}{c_{FP}}$.

Based on observations of past projects' faults and fault removal costs, one could estimate a likely value k_e for the ratio AN/AP and a likely range $[\lambda_l, \lambda_u]$ for λ, to evaluate a given $fp(\underline{z}, t)$ classifier based on the *RoI* identified by

$$y \geq \frac{1-\lambda_l}{\lambda_l} k_e x + 1 - \frac{k_e}{\lambda_l(1+k_e)} \wedge y \geq \frac{1-\lambda_u}{\lambda_u} k_e x + 1 - \frac{k_e}{\lambda_u(1+k_e)} \tag{9}$$

8.2 Cost-reduction *RoIs*

Practically useful *RoIs* should represent strict constraints for defect prediction models. For instance, take $\lambda = 0.9$, i.e., suppose that a false negative is 9 times as expensive as a false positive. The corresponding line described in Formula (8) for the ROC curve in Fig. 4 is

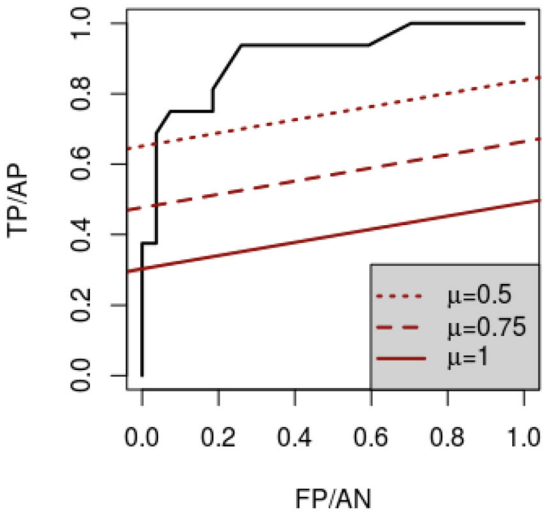


Fig. 6 ROC curve for *fp*(*WMC*) for the `berek` dataset with multiple constraints on normalized cost ($\lambda = 0.9$)

the lowest one in Fig. 6, which does not appear to set a strict constraint and may be of little practical interest.

To obtain a more useful constraint, a software manager may set a maximum acceptable value for the unitary cost, which translates into a maximum value NC_{max} for NC. NC_{max} can be expressed as a fraction μ of $\mathbb{E}[NC_{pop}]$, i.e., $NC_{max} = \mu \frac{k}{(1+k)^2}$. Constraint $NC < NC_{max}$ defines a *RoI* with border

$$y = \frac{1-\lambda}{\lambda} kx + 1 - \mu \frac{k}{\lambda(1+k)} \tag{10}$$

The border depends on parameters k, λ, and μ. We describe in Appendix F the effect of having different values of k, which happens with different datasets. We here investigate the effect of selecting different values of μ, possibly in combination with different values of λ.

Formula (10) shows that μ only influences the intercept of the border. For given values of k and λ, the smaller μ (i.e., the smaller NC_{max}), the higher the border line. Figure 6 shows the borders for the `berek` dataset (which has $k = 27/16$) when $\lambda = 0.9$, for $\mu = 1$, $\mu = 0.75$, and $\mu = 0.5$. Formula (8) is a special case of Formula (10) with $\mu = 1$ and it can be easily shown that, for any given value of μ, Formula (10) describes a pencil of straight lines, one for each value of λ, with center in $(\mu \frac{1}{1+k}, 1 - \mu \frac{k}{1+k})$. Thus, for any given value of μ, different values of λ have the same effect as we described in Section 8.1.

As μ varies, the center point $(\mu \frac{1}{1+k}, 1 - \mu \frac{k}{1+k})$ moves on the straight line $y = -kx + 1$. It moves upwards and to the left as μ decreases, as expected, since the constraint becomes stricter. When μ tends to 0, then the center point tends to (0, 1), which represents perfect classification.

We denote as $RRA(\lambda = c_\lambda, \mu = c_\mu)$ the value of *RRA* obtained for the *RoI* whose border is identified by using $\lambda = c_\lambda$ and choosing $\mu = c_\mu$.

The value of μ is related to the performance of defect prediction models quantified by any metric. For instance, take *Precision* and suppose that a technique for defect prediction

models guarantees a minimum value of *Precision*=c. The border corresponding to *Precision*=c is $y = \frac{c}{1-c}kx$ (see Table 5). This straight line intersects line $y = -kx + 1$ at point $(\frac{1-c}{k}, c)$, which corresponds to $\mu = \frac{(1-c)(1+k)}{k}$. This is the cost reduction proportion that can be obtained with a technique that improves the value of *Precision* from $Precision_{pop}$ to c.

Conversely, suppose we focus on *Precision* and we plan to achieve a μ cost reduction. The required improvement in *Precision* is $c = 1 - \frac{k\mu}{1+k}$.

Similar computations can be carried out for all other performance metrics. The results are in Appendix E.

9 Evaluating *RRA*

RRA is clearly related to *AUC* and Gini's G, so one may wonder if *RRA* is better than *AUC* or G, and, if so, to what extent.

First, the main difference between *RRA*, on one side, and *AUC* and G, on the other side, is that *RRA* assesses a defect proneness model only based on the points of a ROC curve corresponding to defect prediction models that are worth evaluating, while *AUC* and G use all of the points of a ROC curve.

We here assess the extent to which our approach restricts the area of the portion of the ROC space that is taken into account as compared to *AUC* and G, by computing the areas of two types of *RoIs* that we have already used in the previous sections and that we also use in the empirical study of Section 10. The area of *RoI*(*Recall*, *Fall-out*) is equal to $\frac{k}{(k+1)^2}$. As a function of k, this area has maximum value $\frac{1}{4}$, attained for $k = 1$, i.e., $AP = AN$. Thus, the computation of *RRA*(*Recall*, *Fall-out*) takes into account at most only one-fourth of the portion of the ROC space taken into account by *AUC* (whose area is 1), and one-half of the portion taken into account by G (whose area is 0.5). The more k tends to 0 or infinity, the more the area of the *RoI* shrinks. For instance, for $k = \frac{AN}{AP} = 4$, the area of the *RoI* is $\frac{4}{25} = 0.16$, which corresponds to 16% of the area considered for *AUC* and 32% of the area considered by G.

The same phenomenon occurs for the area of *RoI*(*FM*, *NM*), which is equal to $\frac{3k}{(k+2)(2k+1)}$, with maximum value $\frac{1}{3}$ when $k = 1$. For $k = 4$, the value is $\frac{2}{9} \simeq 0.22$, i.e., 22% of the area considered for *AUC* and 44% of the area considered by G.

Thus, these two *RoIs* take into account a rather reduced proportion of the ROC space if compared to *AUC* or even G. In practice, by considering irrelevant regions, *AUC* and G incorporate a large "noise" that is higher for projects with a relatively small (or large) defect density. Section 10 shows some results we obtained for these areas in our empirical study, along with results about the proportion of classifiers of ROC curves that fall into our *RoIs*.

Second, in two cases, the relationships between the values of *RRA* and of the performance metrics *AUC* and G are necessarily strong only for models that are either exceptionally good or exceptionally bad. Suppose that a model is so good as to have $AUC = 1$, then the entire ROC space is under the ROC curve, and both G and *RRA* are therefore equal to 1. The converse is also true. If $RRA = 1$, then the ROC curve is above the entire *RoIs* chosen. Since any *RoI* includes the perfect estimation (0, 1) point, then the ROC curve goes through it, and $AUC = 1$ and $G = 1$. For continuity, exceptionally good models that achieve near-perfect estimation are very likely to have values of *AUC*, G, and *RRA* close to 1. The empirical study in Section 10 provides some evidence on the values of *AUC* for which this relationship exists between *RRA* and the existing metrics *AUC* and G even in approximate form.

When it comes to bad models, the implication is unidirectional, in general. Suppose that a model is so bad as to have $AUC = 0.5$ and therefore $G = 0$, then the ROC curve coincides with the diagonal, which implies that $RRA = 0$ for all kinds of *RRA*. However, $RRA = 0$ does not imply $AUC = 0$ and $G = 0$. As a first example, take the ROC curve in Fig. 5b. We have $RRA(\phi = 0.8) = 0$, while $AUC > 0$. As a second example, suppose that the ROC curve goes through point $(\frac{1}{1+k}, \frac{1}{1+k})$. Suppose that the value of *AUC* is greater than 0—which is true except if the ROC curve entirely coincides with the diagonal. We have *RRA(Recall, Fall-out)*=0. This happens because the ROC curve never enters the region of interest. Thus, for bad models and, as we see in Section 10, for models that are not exceptionally good, *RRA* can provide a different perspective and evaluation than *AUC* and *G*.

Third, *RRA* is customizable, since it allows practitioners and researchers to define the set of points of a ROC curve they are interested in, i.e., those in a *RoI* built by selecting specific performance metrics and reference policies, while this is not possible with *AUC* and *G*.

Fourth, the assessment of whether *RRA* is better than *AUC* or *G* requires defining what "better" actually means in this case. *RRA*, *AUC*, and *G* all provide an overall evaluation of the performance of the defect prediction models $fn(\underline{z},t)$ for all values of t. As such, *RRA*, *AUC*, and *G* are aggregate functions of these models. Thus, to assess them, we would need another aggregate function that provides the "ideal" figure of merit against which we can compare the performance of *RRA*, *AUC*, and *G*. Unfortunately, such "ideal" function does not exist or is not known. Total Cost *TC* would be a suitable "ideal" figure of merit, but it is unknown. That is why metrics like *AUC* and *G* were introduced and used in the first place: they would not have been introduced if *TC* were known. At any rate, *RRA* can take into account different cost models via parameter λ and different cost requirements via parameter μ in ways that *AUC* and *G* can not.

Therefore, the empirical study we present in Section 10 does not and cannot have the goal of showing whether *RRA* is "better" than *AUC* and *G*, or any other metric, for that matter. Rather, we want to show the differences in the assessment of defect proneness models between *RRA* and the existing ones, and show that *RRA* can be more reliable and less misleading.

10 Empirical Study

We analyzed 67 datasets for a total of 87,185 modules from the SEACRAFT repository (https://zenodo.org/communities/seacraft). These data were collected by Jureczko and Madeyski (2010) from real-life projects of different types, and have been used in several defect prediction studies (e.g., (Bowes et al. 2018; Zhang et al. 2017)). The number of modules in the datasets ranges between 20 and 23,014 with an average of 1,300, a standard deviation of 3,934, and a median of 241.

For each module (a Java class, in this case), all datasets report data on the same 20 independent variables.[1] In addition, the datasets provide the number of defects found in each class, which we used to label modules as actually negative and positive. The datasets are fairly different in terms of AP/n ratio, which ranges from 2% to 98.8%. The histogram in Fig. 8 shows the frequency distribution of the proportion of defective modules in the datasets. Though there is a majority of small values in the distributions of n, AP/n, and

[1]Definitions can be found at http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/metric.html.Cyclomatic Complexity appears in the datasets as both the average and the maximum value across the methods of a class.

Table 7 Values of *AUC* and *RRA* for *AUC* ranges for BLR models

AUC		*AUC*	*RRA*(*Recall,Fall-out*)		*RRA*(ϕ=0.4)	
Rank	n	Median	Range	Median	Range	Median
Acceptable	13	0.74	[0.03, 0.26]	0.18	[0.00, 0.00]	0.00
Excellent	29	0.86	[0.27, 0.65]	0.45	[0.00, 0.50]	0.19
Outstanding	25	1.00	[0.35, 1.00]	1.00	[0.02, 1.00]	1.00

LOC—as is for software projects in general—fairly large values are well represented (e.g., half of the projects are larger than 59,000 *LOC*).

For each dataset, we built a BLR model and a Naive Bayes (NB) model using all available measures as independent variables.

Here are the Research Questions that we address in our empirical study.

RQ1 To evaluate defect proneness models in practice, to what extent are the regions of the ROC space used by *RRA* more adequate than the regions used by *AUC* and *G*?

With RQ1, we investigate if, in real-life projects, it is possible to have substantial differences between *RRA* and the existing performance metrics *AUC* and *G*, as these performance metrics take into account different regions of the ROC space and therefore different defect prediction models.

RQ2 How frequently are there substantial differences between *RRA* and traditional performance metrics *AUC* and *G* in using more adequate regions of the ROC space?

By answering RQ2, we check whether *RRA* is truly useful only in corner cases for extreme projects, or for a large share of the population of projects.

10.1 RQ1: Extent of Adequacy of ROC Space Regions Used

For each of the 67 datasets, the BLR model we obtained had a higher *AUC* value than the NB model, with only one exception, for the `xalan 2.7` project. In what follows, we present the results for the 67 BLR models and also discuss the results for the NB model for the `xalan 2.7` project.

To answer RQ1, we consider the interpretation of *AUC* proposed by Hosmer et al. (2013) and illustrated in Table 4. Accordingly, we split the BLR models we obtained into three classes: acceptable, excellent, and outstanding. Note that we obtained only one BLR model with $AUC < 0.7$, specifically, with $AUC = 0.69$. As this value is very close to the 0.7 lower boundary of the acceptable *AUC* category, we include it here in the acceptable *AUC* category of models, instead of analyzing it by itself in a separate category. Then, we computed the values of *RRA*(*Recall, Fall-out*) and *RRA*($\phi = 0.4$): Table 7 provides a summary of *RRA* values for each *AUC* category.

The NB model obtained for the `xalan 2.7` has $AUC = 0.95$, *RRA*(*Recall, Fall-out*)$=0$ and *RRA*($\phi = 0.4$)$=0.01$, while the corresponding BLR model has $AUC = 0.69$ *RRA*(*Recall, Fall-out*)$= 0.03$ and *RRA*($\phi = 0.4$)$= 0$. Note that the values of *RRA* are very low for both models, even though the two models greatly differ in the values of *AUC*. Moreover, *RRA* shows that the apparently outstanding NB model is actually not acceptably accurate.

Table 8 Values of *AUC* and *RRA* for BLR models and *AP/n* ranges

AUC			*AUC*	*RRA*(*Recall,Fall-out*)		*RRA*(ϕ=0.4)	
Range	n	AP/n	median	Range	Median	Range	Median
(0.8, 0.87]	6	l/h	0.86	[0.29, 0.45]	0.39	[0.00, 0.14]	0.07
(0.8, 0.87]	12	mid	0.84	[0.27, 0.52]	0.45	[0.06, 0.32]	0.20
(0.87, 0.9]	5	l/h	0.90	[0.34, 0.54]	0.42	[0.08, 0.30]	0.13
(0.87, 0.9]	6	mid	0.88	[0.51, 0.65]	0.56	[0.29, 0.50]	0.37
(0.9, 1)	9	any	0.92	[0.35, 0.99]	0.74	[0.02, 0.99]	0.55
[1, 1]	16	any	1.00	[1.00, 1.00]	1.00	[1.00, 1.00]	1.00

Let us examine the results of Table 7 for the three categories of *AUC* values. First, models in the acceptable *RRA* category should be rejected: they all have *RRA*($\phi = 0.4$)=0, and quite low *RRA*(*Recall, Fall-out*). Second, models classified as excellent according to *AUC* have a quite large variability of *RRA*. Third, models classified as outstanding according to *AUC* generally have very high values of *RRA*, although exceptions are possible: the model for `jedit 4.3` has $AUC = 0.9$, but also *RRA*(*Recall, Fall-out*)=0.35 and *RRA*(ϕ=0.4)=0.02 (noticeably, `jedit 4.3`'s dataset is characterized by *AP/n*=0.02) and the NB model for `xalan 2.7` even more extreme values.

To better understand the relationship between *AUC* and *RRA*, we split the excellent *AUC* range into two sub-ranges, and also split the resulting model sets according to AP/n. Specifically, we split the model set with excellent *AUC* into those having $AUC \in (0.8, 0.87]$ and those having $AUC \in (0.87, 0.9]$; threshold 0.87 was chosen to have enough models in each subset. As for the 25 models with outstanding *AUC*, 16 were perfect prediction classifiers, with $AUC = 1$. We put them in a separate category from those with *AUC* $\in (0.9, 1)$.

The results are in Table 8, where "mid" indicates values between 0.2 and 0.8 of AP/n, i.e., in the middle part of the [0, 1] interval of AP/n and "l/h" indicates values of AP/n that are either in the low or high range, i.e., less than 0.2 or greater than 0.8. By "any," we indicate that we did not split the models based on AP/n.

Table 8 shows that BLR models with *AUC* in the (0.8,0.87] range and "l/h" values for AP/n mostly have low values of *RRA*: e.g., the median *RRA*($\phi = 0.4$) is just 0.07 for these models. Models with *AUC* in the (0.87,0.9] range and mid values of AP/n have instead higher values of *RRA*. The other models are characterized by variable *RRA*, hence they should be evaluated individually.

As for models with outstanding *AUC* values, Table 8 obviously confirms that, as noted in Section 9, models with *AUC*=1 also have *RRA*=1. The models with *AUC*$\in$ (0.9, 1) have generally high *RRA*(*Recall, Fall-out*) and *RRA*(ϕ=0.4), even though the median of *RRA*(ϕ=0.4)=0.55 is a bit over half the value of *RRA*(ϕ=0.4) of the models with *AUC*=1. There are two important exceptions, as noted above, i.e., the BLR model for `jedit 4.3` with outstanding *AUC*=0.9 and extremely poor *RRA*(ϕ=0.4)=0.02 and the NB model for `xalan 2.7` with outstanding *AUC*=0.95 and extremely poor *RRA*($\phi = 0.4$)=0.01.

To provide additional evidence, Fig. 7 shows the values of *AUC* and *RRA*(ϕ=0.4) for the models obtained from datasets with $AP/n \leq 0.2$ or $AP/n \geq 0.8$.

Together, Table 8 and Fig. 7 indicate that the strong relationship between the values of *AUC* and *RRA* we described in Section 9 only holds when *AUC* is extremely close to 1, but no longer exists for values of *AUC* that are considered excellent or even outstanding.

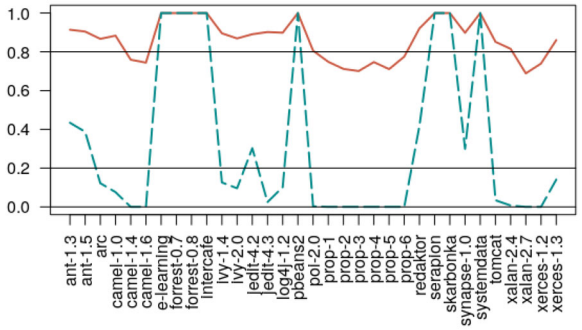


Fig. 7 *AUC*—red continuous line—and $RRA(\phi = 0.4)$—blue dashed line—for projects with $\frac{AP}{n} \leq 0.2$ or $\frac{AP}{n} \geq 0.8$

Thus, in response to RQ1, we can observe that *RRA* appears to take into account a more useful region of the ROC space than *AUC* and *G* in the evaluation of a defect prediction model. As a consequence, *RRA* provides more realistic evaluations than traditional performance metrics, which provide unreliable evaluations, under some conditions.

10.2 RQ2: Frequency of Using more Adequate Regions

To answer RQ2, we need to check whether low or high values of defect density *AP/n* are frequent or rare. Figure 8 shows the distribution of defect density values (rounded to the first decimal) of the projects we considered. Quite a large share (31 projects out of 67) have $AP/n \leq 0.2$, i.e., a fairly small defect density, while 2 out of 67 have $AP/n \geq 0.8$, i.e., a very high defect density.

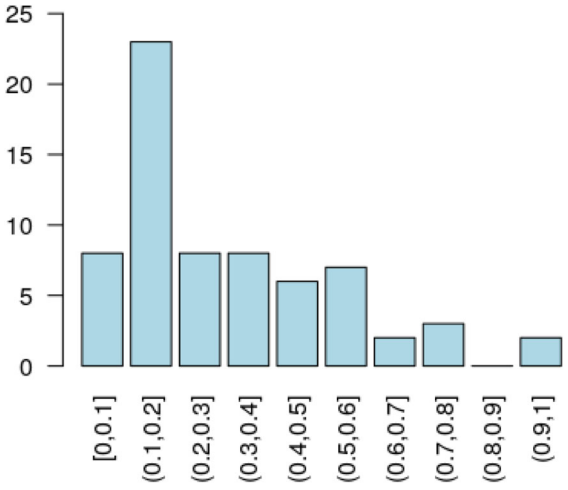


Fig. 8 Distribution of defect density in the datasets by Jureczko and Madeyski

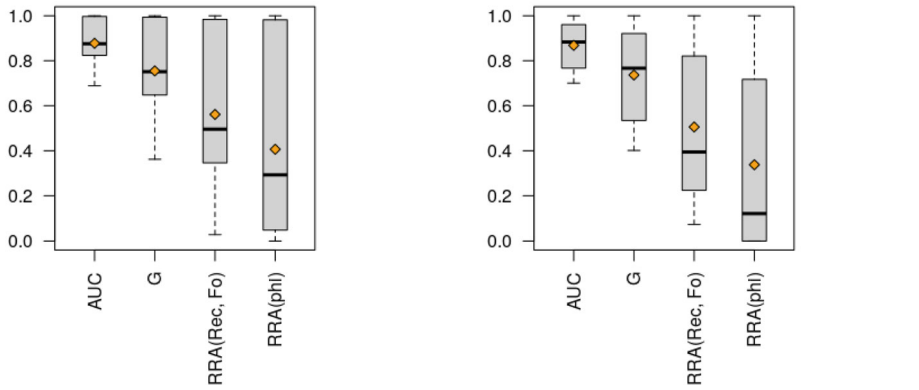


Fig. 9 Comparison of the distributions of *AUC*, *G*, *RRA*(*Recall, Fall-out*) and *RRA*(ϕ=0.4)

Thus, at least for the considered datasets, for about one half of the models, traditional indicators are bound to provide responses based on very large portions of ROC curves where predictions are worse than random.

Also, notice that the range of "l/h" values of AP/n is 40% of the entire AP/n range (i.e., [0,1]), but accounts for 49% of datasets, i.e., projects are more concentrated in these subintervals of AP/n, for which larger portions of *AUC* are meaningless.

To obtain additional quantitative insights, we performed some additional analysis. We computed the areas of *RoI*(*Recall*, *Fall-out*) for the analyzed datasets. We found that the mean area of *RoI*(*Recall*, *Fall-out*) is 0.164, while the median is 0.173, and the standard deviation is 0.067. 49% of datasets have *RoI*(*Recall, Fall-out*) whose area is ≤ 0.16, i.e., more than 84% of the ROC space used to compute *AUC* and 68% of the area above the diagonal considered to compute *G* are in the region representing classifiers that are random or worse than random.

Similarly, we computed the areas of *RoI*(*FM*, *NM*): the mean area is 0.22, the median is 0.24, and the standard deviation is 0.09. 49% of datasets have $RoI(FM, NM) \leq 0.24\overline{2}$, i.e., more than 75% of the ROC space used to compute *AUC* and 52% of the area above the diagonal considered to compute *G* are in the region representing classifiers that are random or worse than random ($0.24\overline{2}$ is the area or *RoI*(*FM*, *NM*) when AP/n is 0.2 or 0.8).

In conclusion, for a large share of datasets, evaluations based on *AUC* or *G* are largely based on regions of the ROC space that should not be considered.

However, one may suppose that the classifiers represented by a ROC curve may be concentrated mostly in *RoIs* such as *RoI*(*Recall*, *Fall-out*). If this is the case, then some of the issues related to using defect prediction models that have performance worse than random policies may be alleviated. Thus, we investigated how many points in the ROC curves we found are outside *RoI*(*Recall*, *Fall-out*). We found that all the models have over 50% of the classifiers out of the *RoI*, and 64% of the models have over 2/3 of the classifiers out of the *RoI*.

Finally, Fig. 9 shows boxplots comparing the distributions of *AUC*, *G*, *RRA*(*Recall, Fall-out*) and *RRA*($\phi = 0.4$). Figure 9a concerns all the models: it can be seen that *AUC* provides quite optimistic evaluations: except for just one case, *AUC* is greater than 0.7, with mean and median well above 0.8. On the contrary, the values of *RRA*(*Recall, Fall-out*) are more

widely spread, indicating that *RRA*(*Recall, Fall-out*) discriminates models more severely and realistically. Values of $RRA(\phi = 0.4)$ are even more widely spread, with lower mean and median. Figure 9b provides the same comparison, considering only the models of the 33 out of 67 datasets having "l/h" AP/n. It is noticeable that the distribution of AUC changes only marginally, with respect to Fig. 9a; on the contrary, the distributions of *RRA*(*Recall, Fall-out*) and $RRA(\phi = 0.4)$ center on lower values, showing that the indications by *AUC* are far too optimistic.

Based on the collected evidence, we can answer RQ2 by stating that *RRA* indicators are useful quite frequently. On the contrary, it appears that *AUC* indications are seldom reliable.

11 The Software Engineering Perspective

The meaning of *RRA* is the same as the meaning of *AUC* and *G*, at a high level: all these indicators provide an evaluation of the performance of a defect proneness model. Accordingly, *RRA* can be applied just like *AUC* and *G*. However, as discussed in Section 7, *RRA* can be adapted to specific needs and goals. For instance, performance evaluation can be based on ϕ or on *FM* and *NM*. Accordingly, the meaning of *RRA* is more specific than the meaning of *AUC* or *G*. We now outline how our proposal can be used during software development.

11.1 Defect Prediction Model Selection

Suppose that the software manager of a software project, e.g., `ivy 2.0`, needs to use a defect prediction model based on several measures. The considered model appears reasonably good, since it has *AUC*=0.87, *RRA*(*FM,NM*)=0.42, *RRA*(*Recall, Fall-out*)=0.39, $RRA(\lambda = 0.9, \mu = 0.5)$=0.22, $RRA(\phi = 0.4)$=0.10.

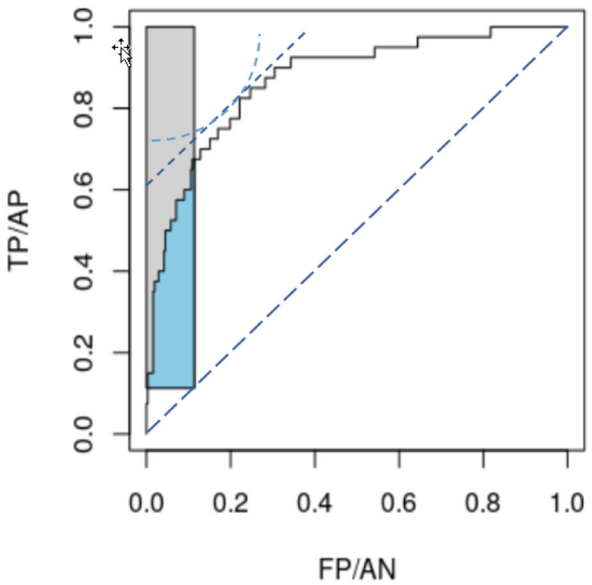


Fig. 10 ROC curve and *RoI*(*Recall*, *Fall-out*) of the LOC-based model for `ivy 2.0`

Figure 10 shows the corresponding ROC curve and *RoI*(*Recall, Fall-out*). When it comes to choosing a defect proneness threshold to build a defect prediction model, the software manager realizes that

1. by selecting the models corresponding to points close to (0,1), i.e., those on the dotted curve in Fig. 10, as is often suggested, one obtains a model with *Fall-out* worse than random estimations;
2. by selecting the model corresponding to the tangent point on the highest isocost line that touches the curve, i.e., those touched by the dashed line in Fig. 10, (as suggested in Powers (2011), for instance), one chooses a model whose *Fall-out* is worse than random estimations.

Thus, conventional wisdom concerning the position of the best fault prediction model in the ROC space is not always reliable. On the contrary, the *RoI* highlighted in Fig. 10 suggests where useful thresholds should be chosen from.

11.2 Defect Proneness Model Selection

Suppose that a software manager has two defect proneness models (for instance, built with different modeling techniques) and needs to decide which one to use. *AUC* provides a spurious assessment of the performance of the defect proneness models, because it is based even on parts of the ROC space that are not of interest for the software manager. As Fig. 1b shows, two ROC curves $ROC_1(x)$ and $ROC_2(x)$ can intersect each other in such a way that the final selection is based mostly on parts of the ROC space that should not be considered.

However, by focusing only on the *RoI*, the software manager may find that $ROC_1(x)$ is predominantly (or even always) above $ROC_2(x)$ in the *RoI*, so the defect proneness model corresponding to $ROC_1(x)$ should be preferred to the one corresponding to $ROC_2(x)$ when building a defect prediction model. Thus, the software manager can make a better informed (and even easier) decision as to which defect proneness model to use.

11.3 Assessing Costs and Benefits of Additional Measures

Suppose a software manager is in charge of a software project in which no real systematic code measurement process is in place, but only data on modules' size (expressed in *LOC*) and defectiveness are currently available. The software manager builds a *LOC*-based defect proneness model *fp*(*LOC*) and uses *AUC* to decide whether the model's performance is good enough. For instance, suppose this was the case of the `xerces 1.4` project. The value of *AUC* for *fp*(*LOC*) is 0.75, which is right in the middle of the acceptable range. However, if the project manager also checks performance with the *RRA* metrics, the values *RRA*(*Recall, Fall-out*) $= 0.2$ and $RRA(\phi = 0.4) = 0.0006$ show that *fp*(*LOC*) actually has much poorer performance that *AUC* would indicate.

Thus, the project manager may decide that more measures are needed to build better performing models and start a systematic code measurement collection process and finally get a *fp* model based on multiple code measures. Suppose that the total set of measures obtained after implanting the new measurement process is the one in the datasets by Jureczko and Madeyski. The BLR model that uses all of them has very high *RRA*(*Recall, Fall-out*)=0.77 and $RRA(\phi = 0.4)$=0.73. Thus, this model has much better performance than the *LOC*-based model and therefore higher trustworthiness.

However, there may be costs associated with establishing a systematic measurement program, which need to be weighed against the benefits of having better performing models.

For instance, if additional measures can be had by building or buying an automated tool that analyzes software code, the software manager incurs a one-time cost. If, instead, the collection itself of the measures requires the use of resources, then there is a cost associated with every execution of the measurement program. As an example, suppose that a well-performing model requires the knowledge of the Function Points associated with the software system. Counting Function Points can be quite expensive (Jones 2008; Total Metrics 2007). Thus, by using *RRA* -based metrics, the software manager can have a better assessment of the costs and benefits due to an expanded measurement program.

11.4 Adaptability to Goals

Unlike with *AUC* and *G*, software managers can "customize" *RRA* for their projects and goals. The value of λ to be used in $RRA(\lambda, \mu)$ is derived from the unit costs c_{FN} and c_{FP}. Thus, $RRA(\lambda, \mu)$ depends on the characteristics of a project, as different projects have different values of c_{FN} and c_{FP} and therefore of λ. So, $RRA(\lambda, \mu)$ takes into account costs more precisely than *AUC* and *G* can for a specific project.

Parameter μ is related to the project goals, since it is the desired proportion of unitary cost reduction that determines the maximum unitary cost (see Section 8.2). Using $RRA(\lambda, \mu)$ allows software managers to restrict the selection of a defect prediction model only among those defect proneness models for which $RRA(\lambda, \mu) > 0$. This kind of selection cannot be carried out by using *AUC* and *G*, which may actually be misleading, as shown in Section 10.

In addition, in Section 8.2, we showed that there is a relationship between performance metrics and the cost reduction proportion μ that can be achieved. Poorly performing models imply low levels of cost reduction and, conversely, high levels of cost reduction imply the need for well-performing models. This may call for building better models, as shown, for instance, in Section 11.3.

Software Defect Prediction researchers can use our proposal to have a more precise assessment of the quality of defect prediction models. Like software managers, they can also customize *RRA* for their goals. For instance, they can use $RRA(\phi)$ to have a general assessment of models or focus on specific performance metrics by using *RRA*(*FM,NM*), for instance, or any other ones. The variety of ways in which *RRA* can be defined and used goes beyond what has already been defined and used to delimit the part of the ROC curve to consider in other fields such as medical and biological disciplines (see the review of the literature in Section 13.1).

12 Threats to Validity of the Empirical Study

We here address possible threats to the validity of our empirical study, which we used to demonstrate our proposal and provide further evidence about it.

Some external validity threats are mitigated by the number of real-life datasets and the variety in their characteristics such as application domains, $\frac{AP}{n}$ ratio, number of modules, and size, as indicated in Section 10.

The values of *AUC* and *G* are computed, according to common practice, based on the training set used to build defect prediction models; similarly, to compute *RRA*, we took the training set as the test set too. Using a different test set than the training set may change the value of $\frac{AP}{n}$. This concept drift would affect the *RoIs* to be taken into account (e.g., *RoI*(*Recall*, *Fall-out*)), and, therefore, the value of *RRA* (e.g., *RRA*(*Recall*, *Fall-out*)). Thus, we may have obtained different results with different test sets than the training sets.

As for the construct validity of *RRA*, based on Sections 5—8, Section 9 shows that *RRA* specifically addresses some of the construct validity issues related to *AUC* and *G*, which are widely used in Software Defect Prediction and several other disciplines.

Construct validity may be threatened by the performance metrics used. For instance, *FM* has been widely used in the literature, but it also has been largely criticized (Shepperd et al. 2014). We also used *Precision*, *Recall*, *Specificity*, *NPV*, *NM*, and ϕ, to have a more comprehensive picture and set of constraints based on different perspectives. At any rate, our approach is not limited to any fixed set of performance metrics, and any other may be used as well, as long as it is based on confusion matrices.

Also, we used BLR and NB because of the reasons explained in Section 10. Other techniques may be used, but the building of models is not the goal of this paper: we simply needed models for demonstrative purposes.

13 Related Work

Given the importance of defect prediction in Software Engineering, many studies have addressed the definition of defect proneness models. They are too many to mention here. ROC curves have been often used to evaluate defect proneness models, as reported by systematic literature reviews on defect prediction approaches and their performance (Arisholm et al. 2010; Beecham et al. 2010a; Hall et al. 2012).

There has been an increasing interest in ROC curves in the Software Defect Prediction and, more generally, Empirical Software Engineering literature in the last few years. For instance, 82 papers using ROC curves appeared in in the 2007-2018 period in three major Software Engineering publication venues, namely, "IEEE Transactions on Software Engineering," "Empirical Software Engineering: an International Journal," and the "International Symposium on Empirical Software Engineering and Measurement," while no papers using ROC curves appeared before 2007.

We here first describe and discuss proposals for performance metrics that have appeared in the general literature on ROC curves analysis, to address some of the issues related to the adoption of *AUC* (Section 13.1).

Then, we review a few of the related works published in the Empirical Software Engineering literature, to provide an idea of what kind of work has been done with ROC curves (Section 13.2).

Also, we show in Section 13.3 how cost modeling can be addressed by our approach even with different cost models than the one we use in Section 8.

13.1 ROC Curve Performance Metrics

A few proposals define performance metrics that take into account only portions of a ROC. These approaches define various forms of a partial *AUC* metric (*pAUC*), which has also been implemented in the R package `pROC`, available at `https://web.expasy.org/pROC/` (Robin et al. 2011).

McClish (1989) computes *pAUC* as the area under $ROC(x)$ in an interval $[x_1, x_2]$ between two specified values of x. To compare the performance of digital and analog mammography tests, Baker and Pinsky (2001) compare the partial *AUCs* for the two different ROC curves in an interval between two small x values. For mammography-related applications too, Jiang et al. (1996) propose a different version of partial AUC, in which they only take into account the high-recall portion of the ROC space, in which $y > \bar{y}$, where $\bar{y}$ is a specified value of

y. They define a metric as the ratio of the area under $ROC(x)$ and above $y = \bar{y}$ to the area above $y = \bar{y}$, i.e., $1 - \bar{y}$. Dodd and Pepe (2003) introduce a nonparametric estimator for partial AUC, computed based on an interval of x, like in McClish (1989), or based on an interval of y. All four papers carry out further statistical investigations (e.g., the definition of statistical tests for comparing the areas under different ROC curves), based on statistical assumptions.

McClish also defines a "standardized" version of $pAUC$ that takes into account only the part of the vertical slice in the $[x_1, x_2]$ interval that is also above the diagonal, i.e., the trapezoid delimited by the diagonal $y = x$, the vertical lines $x = x_1$ and $x = x_2$, and the horizontal line $y = 1$. Specifically, the standardized metric is based on the ratio (which we call here pG) between, on one hand, the area under the curve and above the diagonal and, on the other hand, the area of the trapezoid. The standardized metric is then defined as $\frac{1}{2}(1 + pG)$. Note that pG coincides with the partial Gini index that was defined along the same lines in Pundir and Seshadri (2012), by computing the normalized value of Gini's G in an interval $[x_1, x_2]$, and was used in Lessmann et al. (2015) with x in the interval $[0, 0.4]$. Like between AUC and G, there is a relationship between $pAUC$ and pG. It can be easily shown analytically that $pAUC = pG\,(1 - \bar{x}) + \bar{x}$, where $\bar{x} = \frac{x_1 + x_2}{2}$ is the midpoint of the $[x_1, x_2]$ interval.

These approaches and ours share the idea that the evaluation of $ROC(x)$ can be of interest with respect to portions of the curve. However, the portion of the ROC space taken into account is either a vertical slice or trapezoid or a horizontal slice of the ROC space, and does not take different forms, like the ones that are possible with our approach, e.g., the ones depicted in Figs. 3a and 5. Also, these portions of the ROC space are not necessarily *RoIs* according to Definition 3. Thus, there are some *RoIs* that are not taken into account by these approaches, and *vice versa*. The reason lies in the goals of these proposals and ours. Our goal is to identify the classifiers whose performance is better than some reference values. The other approaches aim to limit the set of interesting values of either *Recall* or *Fall-out*. So, they may take into account classifiers whose performance is worse than reference values, e.g., those obtained via random classifiers.

The meaning of AUC has also been investigated by Hand (2009), who finds that computing AUC is equivalent to computing an average minimum misclassification cost with variable weights. Specifically, Hand finds that the expected minimum misclassification cost is equal to $2\frac{AP \cdot AN}{n^2}$(1-AUC) when the values of c_{FN} and c_{FP} are not constant, but depend on the classifier used. This poses a conceptual problem, since c_{FN} and c_{FP} should instead depend on the software process costs and the costs related to delivering software modules with defects, and not on the classifier used. Our proposal partially alleviates Hand's issue, by delimiting and reducing the set of classifiers taken into account, so the variability of the values of c_{FN} and c_{FP} is reduced. Hand also defines an alternative metric, H, which, however, relies on two assumptions: (1) $c_{FN} + c_{FP}$ and $\lambda = \frac{c_{FN}}{c_{FN}+c_{FP}}$ are statistically independent; (2) the probability density function of λ used for the computation of $\mathbb{E}[TC]$ is a Beta distribution with specified parameters. As for the second assumption, Hand discards the use of a uniform distribution because it would treat extreme and more central values of λ as likely. Hand also advocates the use of different functions that can be more closely related to practitioners' and researchers' goals, if available. Hand shows how H can be estimated. Given the correspondence between straight lines and cost models, the identification of a *RoI* with our proposal may help delimit the set of the values of λ to take into account.

Other performance metrics for ROC curves are described in Swets and Pickett (1982). These metrics are all based on a binormal ROC space, in which the abscissa and the ordinate represent the normal deviates obtained from x and y. One of the recommended metrics is

A_z, which is the area under the curve of a transformed ROC curve in the binormal ROC space. Since a border can be transformed into a line in the binormal ROC space, our *RoIs* can help take into account only the portion of the transformed ROC curve that is relevant, and compute only the area under that portion. Other metrics are d', d'_e, and Δm, which are generalized by d_a, which represents the distance of a transformed ROC curve from the origin of the binormal ROC space. Also, Swets and Pickett (1982) mention metric β, which can be computed based on costs and benefits of positive and negative observations, which can only be subjectively assessed.

Papers (Flach 2003; Provost and Fawcett 2001; Vilalta and Oblinger 2000) define "iso-performance lines" or "isometrics," i.e., those ROC space lines composed of classifiers with the same value for some specified performance metric. Our proposal uses those lines as borders for *RoIs* and shows how to derive them starting from random policies, to delimit *RoIs*.

13.2 ROC Curves and *AUC* in Empirical Software Engineering

ROC curves have been used in Empirical Software Engineering for the assessment of models for several external software attributes (Fenton and Bieman ; Morasca 2009).

Here are just a few recent examples of the variety of ways in which ROC curves have been used to assess defect prediction models: Di Nucci et al. (Nucci et al. 2018) use ROC curves and *AUC* for models based on information about human-related factors; McIntosh and Kamei (2018) for change-level defect prediction models; Nam et al. (2018) for heterogeneous defect prediction; Herbold et al. (2018) to assess the performance of cross-project defect prediction approaches.

As for other external software attributes: Kabinna et al. (2018) use ROC curves and *AUC* to assess the change proneness of logging statements; da Costa et al. (2018) to study in which future release a fixed issue will be integrated in a software product; Murgia et al. (2018) to assess models for identifying emotions like love, joy, and sadness in issue report comments; Ragkhitwetsagul et al. (2018) to evaluate code similarity; Arisholm et al. (2007) to assess the performance of predictive models obtained via different techniques to identify parts of a Java system with a high probability of fault; Dallal and Morasca (2014) to evaluate module reusability models; Posnett et al. (2011) to study the risk of having fallacious results by conducting studies at the wrong aggregation level; Cerpa et al. (2010) to evaluate models of the relationships linking variables and factors to project outcomes; Malhotra and Khanna (2013) to assess change proneness models.

ROC curves (with and without *AUC*) have also been used in Empirical Software Engineering studies to find optimal thresholds t for $fn(\underline{z},t)$ to build a defect prediction model. For instance, Shatnawi et al. (2010) use *AUC* to quantify the strength of the relationship between a variable z and defect proneness. The threshold selected by Tosun and Bener (2009) corresponds to the ROC curve point at minimum Euclidean distance from the ideal point (0, 1), which represents perfect estimation. The threshold selected by Sánchez-González et al. (2012) corresponds to the farthest point from the ROC diagram diagonal (see also Mendling et al. (2012)).

13.3 Cost Modeling

The cost related to the use of defect prediction models has been the subject of several studies in the literature that focused on misclassification costs (Hand 2009; Jiang and Cukic 2009; Khoshgoftaar and Allen 1998; Khoshgoftaar et al. 2001; Khoshgoftaar and Seliya 2004), or

used cost curves (Drummond and Holte 2006; Jiang et al. 2008). A recent paper (Herbold) defines a cost model based on the idea that a defect may affect several modules and a module may be affected by several defects. Herbold's cost model also allows the use of different Verification and Validation costs for different modules, different costs for different defects, and different probabilities that Verification and Validation activities miss a defect.

Here we provide a detailed discussion for the cost model investigated by Zhang and Cheung (2013). Specifically, Zhang and Cheung use the overall cost of a prediction model $C_p = c_{FP}(TP + FP) + c_{FN}FN$, which includes $c_{FP}TP$, and derive two inequalities that must be satisfied by the confusion matrix of a defect prediction model. We now show how this cost model can be studied in our approach, by defining the inequalities proposed by Zhang and Cheung as *RoI* borders.

The first inequality is derived by comparing the value of C_p obtained with a binary classifier and the value obtained by trivially estimating all modules positive. When $\lambda \neq \frac{1}{2}$, the equation of the first border is

$$y - 1 = \frac{(1-\lambda)k}{2\lambda - 1}(x - 1) \tag{11}$$

This is a pencil of straight lines going through point (1, 1). A straight line from this pencil defines an effective border (i.e., the straight line is in the upper-left triangle) if and only if its slope is between 0 and 1, i.e., $0 < \frac{(1-\lambda)k}{2\lambda-1} < 1$. When $\lambda > \frac{1}{2}$, the slope is nonnegative and it can be shown that the slope is less than 1 if and only if $\frac{c_{FP}}{c_{FN}} < \frac{AP}{n}$. When $\lambda < \frac{1}{2}$, the slope is negative, and the straight lines are outside the ROC space. When $\lambda = \frac{1}{2}$, the equation of border is $x = 1$, which is not an effective border in the ROC space.

The second inequality is derived by comparing the value of C_p obtained with a defect prediction model and the value obtained with the uni random policy. The second inequality, however, turns out to always set the diagonal $y = x$ as the border straight line, so it does not introduce any real constraints.

At any rate, any cost model based on the cells of the confusion matrix related to a defect prediction model can be dealt with by our approach.

14 Conclusions and Future Work

In this paper, new concepts and techniques are proposed to assess the performance of a given defect proneness model $fp(\underline{z})$ when building defect prediction models. The proposed assessment is based on two fundamental concepts: 1) only models that outperform reference ones—including, but not limited to, random estimation—should be considered, and 2) any combination of performance metrics can be used. Concerning the latter point, not only do we allow researchers and practitioners to use the metric they like best (e.g., *F-measure* or ϕ), but we introduce the possibility of evaluating models against cost, and we show that there is a clear correspondence between performance metrics (such as *Recall*, *Precision*, *F-measure*, etc.) and cost.

Using the proposed technique, practitioners and researchers can identify the thresholds worth using to derive defect prediction models based on a given defect proneness model, so that the obtained models perform better than reference ones. Our approach helps practitioners evaluate competing defect prediction models and select adequate ones for their goals

and needs. It allows researchers to assess software development techniques based only on those defect prediction models that should be used in practice, and not on irrelevant ones that may bias the results and lead to misleading conclusions.

Unlike the traditional *AUC* metric, which considers the entire ROC curve, our approach considers only the part of the ROC curve where performance—evaluated via the metrics of choice—is better than reference performance values, which can be provided by reference models, for instance.

We show that *RRA*—when used with suitable areas of interest, like those that exclude random behaviors—is theoretically sounder than traditional ROC-based metrics (like *AUC* and Gini's *G*). The latter are special cases or *RRA*, but computed on areas that include worse than random classifiers.

We also applied *RRA*, *G*, and *AUC* to models obtained from 67 real-life projects, and compared the obtained indications. *RRA* appeared to provide much deeper insight into the actual performance of models.

RRA proved more adequate than *AUC* and *G* in capturing the information used in the evaluation of defect prediction models. Specifically, *AUC* and *G* appeared to consider a large amount of information pertaining to random (and worse) performance conditions. As a consequence, *AUC* and *G* often reported high performance levels, while the performance of the corresponding models was much lower. In these cases, *RRA* provided much more realistic indications, revealing these low performance levels. Our analysis showed also that *AUC* and *G* can be quite frequently misleading.

Although in the empirical validation (Section 10) only measures taken on modules were used as independent variables, other types of measures—e.g., process measures—could be used in exactly the same way. That is, our approach is applicable to a broader class of models than those considered in this paper.

As a further generalization, our approach can be used outside Software Defect Prediction. What is needed is a scoring function, so that a ROC curve can be built. Thus, if such a model for, say, availability is known, it can be used in our approach in exactly the same way as defect proneness models.

Even more generally, the approach can be conceptually used for any kind of scoring function, so it can be used in disciplines beyond Empirical Software Engineering. Also, the approach can be used with other kinds of constraints that can be set on scoring function. For instance, one can also set a constraint on the value of the first derivative of the scoring function, as we did in our previous work (Morasca and Lavazza 2017) to define risk-averse thresholds for defect proneness models.

Future work will be needed to provide more evidence about the usefulness and the limitations of the approach, including

- the assessment of the approach on more datasets
- the use of additional performance metrics, to have a more complete idea of the performance of the classification
- a more in-depth study of the characteristics of *RRA*, for instance, by introducing statistical tests to check whether the differences in the values of *RRA* of different ROC curves are statistically significant
- the investigation of other techniques for obtaining an overall assessment of a defect proneness model
- the investigation of other cost models, such as the one recently introduced by (Herbold)

– the application to other external attributes that can be quantified by means of probabilistic models (Krantz et al. 1971; Morasca 2009), e.g., maintainability, usability, reusability.

Acknowledgements This work has been partially supported by the "Fondo di ricerca d'Ateneo" of the Università degli Studi dell'Insubria.

Funding Information Open access funding provided by Università degli Studi dell'Insubria within the CRUI-CARE Agreement.

Appendix A: Proof of Properties of *RoIs*

We here prove the properties of *RoIs* that we introduced in Section 6.

Property 1 Intersection of RoI.
The intersection of any number of *RoIs* is a *RoI*.

Proof Let R_1 and R_2 be two *RoIs* and take $(x, y) \in R_1 \cap R_2$. We need to show that $ULR(x, y) \subseteq R_1 \cap R_2$. By contradiction, suppose that $\exists (x', y') \in ULR(x, y)$ such that $(x', y') \notin R_1 \cap R_2$. This is not possible, since $(x', y') \in ULR(x, y)$, which implies that $(x', y') \in R_1 \wedge (x', y') \in R_2$. So, $(x', y') \in R_1 \cap R_2$. This proof can be easily extended in the case of the intersection of any number of *RoIs*. □

Property 2 Upper-left Rectangle.
The smallest *RoI* to which a point (x, y) in the ROC space belongs is $ULR(x, y)$.

Proof Suppose, by contradiction, that, given point (x, y), R is the smallest *RoI* to which (x, y) belongs, i.e., none of its proper subsets $R' \subset R$ that include (x, y) is a *RoI*. However, $R \cap ULR(x, y)$ is a *RoI* because of Property 1 and $R \supseteq ULR(x, y)$ because of the definition of *RoI*. Thus, $R \cap ULR(x, y) = ULR(x, y) \subseteq R$, so $ULR(x, y)$ is not larger than any other *RoIs* R. Because of the definition of *RoI*, no proper subset of $ULR(x, y)$ is a *RoI*. □

Property 3 Perfect Estimation Point.
A *RoI* always contains point (0, 1).

Proof Point (0, 1) is in the upper-left rectangle of all points of the ROC space, so it must be in any *RoI*. □

Property 4 Intersection of RoIs is Nonempty.
The intersection of any number of *RoIs* is nonempty.

Proof This is a direct consequence of Property 3, because all *RoIs* share at last point (0, 1). □

Property 5 Connectedness of a RoI.
A *RoI* is connected.

Proof This property is a consequence of Property 4. By contradiction, let $R = R_1 \cup R_2$ be a *RoI* composed of two unconnected *RoIs* R_1 and R_2. Thus, $R_1 \cap R_2 = \emptyset$, which is not possible, as shown by Property 4. □

Property 6 Monotonic Border.
The border of a *RoI* is a (non necessarily monotonically) increasing function.

Proof By definition, the border of a *RoI* is continuous, since a *RoI* is a connected subset (see Property 5). Thus, the border can be represented by means of two continuous parametric functions $x = x(s)$ and $y = y(s)$, where s plays the role of the parameter.

We first show that the border is a single-valued function, i.e., the border does not bend in such a way as to have a "U-turn". More formally, we prove that there are no three values s', s'', and s''' such that $s' < s'' < s'''$ and, at the same time, $x(s') = x(s''')$ and $y(s') = y(s''')$ and $y(s') \neq y(s') \neq y(s''')$. For instance, after it goes through point $(x(s'), y(s'))$ and then through point $(x(s''), y(s''))$ to the right of point $(x(s'), y(s'))$, it does not go through point $(x(s'), y(s'''))$ with the same x-value as $(x(s'), y(s'))$ but with a different y-value, which is to the left of point $(x(s''), y(s''))$. Suppose that $x(s') < x(s'')$ and that $y(s') < y(s''')$. Then, there are points of the ROC space in $ULR(x(s''), y(s''))$ that do not belong to the *RoI*, which contradicts Property 2. So, the border is a single-valued function.

Suppose now that, by contradiction, the border $B(x)$ is not an increasing function. Suppose, therefore, that there exist x_1 and x_2 with $x_1 < x_2$ such that $B(x_1) > B(x_2)$. Then, not all points in $ULR(x_2, b(x_2))$ belong to the *RoI* of $(x_2, B(x_2))$, which contradicts Property 2. So, the border is an increasing function. □

Appendix B: Computation of Performance Metrics

We here show how to compute the performance metrics of Table 2 in terms of x and y.

First, they can all be built as functions of the ratios of the cells of a confusion matrix (*TP*, *FP*, *TN*, and *FN*) to the row and column totals (*AP*, *EP*, *AN*, and *EN*). This is true by definition for the metrics that assess performance with respect to the positive class *Precision* and *Recall*, and therefore for *F-measure* too. Likewise, this is true by definition for the metrics that assess performance with respect to the negative class *NPV* and *Specificity*, and therefore for *NM* too. As for overall metrics, this is the case for *J* and *Markedness* by definition, so it is also the case for ϕ.

Now, we show how to express these ratios in terms of x and y. We start with the ratios of the cells to AP: $\frac{TP}{AP} = y$; $\frac{FP}{AP} = \frac{FP}{AN}\frac{AN}{AP} = kx$; $\frac{TN}{AP} = \frac{AN-FP}{AP} = k - \frac{FP}{AP} = k(1-x)$; $\frac{FN}{AP} = \frac{AP-TP}{AP} = 1 - y$.

Via an example, we show how the other ratios can now be derived. Ratio $\frac{TP}{EP}$ is clearly equal to $\frac{TP}{AP}\frac{AP}{EP}$. Now, $\frac{AP}{EP} = \frac{AP}{FP+TP} = \frac{1}{\frac{FP}{AP}+\frac{TP}{AP}} = \frac{1}{kx+y}$. Thus, $\frac{TP}{EP} = \frac{y}{kx+y}$. Likewise, we can compute all other ratios whose denominator is EP, by multiplying the corresponding ratios obtained with AP at the denominator by $\frac{1}{kx+y}$. As for the ratios with denominator AN, we multiply the corresponding ratio obtained with AP as the denominator by $\frac{AP}{AN} =$

$\frac{1}{k}$. Finally, for the ratios whose denominator is EN, we multiply the corresponding ratio obtained with AP as the denominator by $\frac{AP}{EN} = \frac{AP}{TN+FN} = \frac{1}{\frac{TN}{AP}+\frac{FN}{AP}} = \frac{1}{k(1-x)+(1-y)}$.

Appendix C: The Behavior of Border Straight Lines

We discuss how the borders obtained for each performance metric change depending on k, c, and p (see Table 5).

Precision. The general straight line $y = \frac{c}{1-c}kx$ goes through the origin and has a slope that is proportional to k and increases with c. Thus, the higher k and/or c, the stricter the constraint, i.e., the smaller the *RoI* , as expected. The *uni* straight line is $y = x$ regardless of p, so it is also the *pop* straight line. This is the diagonal of the ROC space, so it does not provide any additional constraints on the values of x and y, as only ROC curves in the upper-left half of the ROC space should be considered, as explained in Section 5.1.

Recall. The general straight line is horizontal. As expected, the constraint becomes stricter as c increases. Unlike with *Precision*, k has no influence on the strictness of the constraint (unless c itself is a function of k, like in the *pop* case). The *uni* straight line shows that the higher p, the stricter the constraint. The *pop* straight line can be rewritten as $y = \frac{AP}{n}$.

F-measure. The general straight line intersects the x-axis at $x = -\frac{1}{k}$ and the y-axis at $y = \frac{c}{2-c} \leq 1$. Both the intercept and the coefficient of x increase with c, and the coefficient also increases with k. As for the *uni* straight line, given p, the larger k, the larger the slope and the intercept, so the stricter the constraint. Also, given k, the larger p, the larger the slope and the intercept, so the stricter the constraint. The *uni* line intersects the vertical line $x = 1$ at $y = \frac{pk+p}{pk+1} \leq 1$, so it extends across the entire horizontal span of the ROC space. In the *pop* straight line, when k increases from 0 to ∞, the slope monotonically increases from 0 to 1 and the intercept monotonically decreases from $\frac{1}{2}$ to 0. Thus, when $k = 0$, we have the horizontal line $y = \frac{1}{2}$, which, as k increases, tilts and tends to the diagonal $y = x$ when $k \to \infty$.

NPV. The general straight line goes through point $(1, 1)$ and has a slope proportional to k and an intercept that decreases with k. Thus, the higher k, the less strict the constraint. Also, the slope decreases and the intercept increases when c increases, i.e., the constraint becomes stricter as the minimum acceptable value of *NPV* increases. The *uni* (and therefore *pop*) straight line is $y = x$, like in the *Precision* case.

Specificity. The general straight line is vertical. As expected, the constraint becomes stricter as c increases. Unlike with *NPV*, k has no influence on the strictness of the constraint (unless, again, c itself is a function of k, like in the *pop* case). The *uni* straight line shows that the higher p, the stricter the constraint. The *pop* straight line can be rewritten as $x = \frac{AP}{n}$.

NM. In the general straight line, given a value of c, the coefficient of x increases with k and the intercept decreases. Given a value of k, the coefficient of x decreases with c and the intercept increases. As for the *uni* straight line, when p is fixed, the higher k, the steeper the slope, but the lower the intercept. When k is fixed, the slope increases when p increases, but the intercept decreases. In both cases, the constraint becomes less strict. The straight line has a negative intercept $-\frac{p}{1-p}k$, it intersects the horizontal line $y = 0$ at $x = \frac{pk}{1-p+k} \leq 1$, and the horizontal line $y = 1$ at $x = \frac{1-pk}{1-p+k} \leq 1$, so it extends across the entire vertical span of the ROC space. With any random policy, when $k = 0$ the

straight line is the diagonal $y = x$, and when $k \to \infty$, the straight line tends to become the vertical straight line $x = p$.

In the *pop* straight line, p and k are no longer independent, so the results on the behavior of slope and intercept when k varies while p is fixed no longer hold. Instead, with *pop*, the slope increases with k and the intercept remains constant.

Appendix D: Conic Sections as Borders

Here, we show that the border equations for *Markedness* (see Appendix D.1) and ϕ (see Appendix D.2) of Table 6 are conic sections, and discuss some of their properties.

Recall that a conic section can be analytically represented via the quadratic equation (which we describe with doubled terms for mathematical convenience)

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0 \tag{12}$$

which represents an ellipse if and only if $B^2 - AC < 0$, a parabola if and only if $B^2 - AC = 0$, and a hyperbola if and only if $B^2 - AC > 0$.

D.1 Markedness

The border equation for *Markedness* in Table 6 is, in implicit form,

$$ck^2x^2 + 2ckxy + cy^2 - (ck^2 + ck + k)x - (ck - k + c)y = 0 \tag{13}$$

It is immediate to show that $B^2 = AC$. Thus, the curve in Formula (13) is a parabola. The following properties of the parabola can be proven.

- There is one degenerate case that we deal with immediately, which occurs when $c = 0$. The parabola of Formula (13) degenerates into the straight line $y = x$, i.e., the diagonal. From this point on, we therefore assume that $c > 0$.
- The symmetry axis of the parabola is

$$y = -kx + \frac{1+k}{2} - \frac{k(1-k)}{2c(1+k^2)} \tag{14}$$

- The directrix of the parabola is

$$y = \frac{1}{k}x + \frac{\left(c\frac{1+k^2}{k} + \frac{k-1}{1+k}\right)^2 + 1}{4c\frac{1+k^2}{1+k}} \tag{15}$$

It is immediate to show that the parabola goes through the origin, which is below the directrix. Since parabolas never intercept their directrices, the entire parabola of Formula (13) lies below the directrix. Therefore, the vertex of the parabola lies below the directrix too and the parabola lies to the "south-east" of its vertex.

D.2 Ellipses as Constant ϕ Curves

Formula (16), already shown in Table 6, describes the curve obtained for $\phi = c$

$$(k + c^2k^2)x^2 + 2(c^2k - k)xy + (c^2 + k)y^2 - c^2(k^2 + k)x - c^2(k + 1)y = 0 \tag{16}$$

This curve is an ellipse, since $B^2 - AC < 0$, i.e.,

$$(c^2k - k)^2 - (k + c^2k^2)(c^2 + k) = -c^2k(k + 1)^2 < 0 \tag{17}$$

The following properties of the ellipse can be proven.

- The ellipse intercepts the boundaries of the ROC space at (0, 0), $(0, \frac{c^2(k+1)}{c^2+k})$, $(\frac{1-c^2}{1+c^2k}, 1)$, (1, 1), $(1, \frac{(1-c^2)k)}{c^2+k})$, $(\frac{c^2(k+1)}{1+c^2k}, 0)$.
- The ellipse is centered in point $(\frac{1}{2}, \frac{1}{2})$, in the center of the ROC space
- The equation of the major axis of the ellipse is

$$y = \frac{-c^2(1-k^2) + \sqrt{c^4(1-k^2)^2 + 4k^2(c^2-1)^2}}{2k(1-c^2)}\left(x - \frac{1}{2}\right) + \frac{1}{2} \tag{18}$$

- Regardless of the value of k, when $c^2 \to 0$, the slope tends to +1, i.e., the major axis tends to the diagonal of the ROC space. When $c^2 \to 1$, the slope tends to 0, i.e., the major axis tends to the horizontal straight line $y = \frac{1}{2}$.
- When $k < 1$, the slope of the major axis is a monotonically decreasing function of c^2 and when $c^2 \to 1$, the slope tends to 0, i.e., the major axis tends to the horizontal straight line $y = \frac{1}{2}$.
- When $k > 1$, the slope of the major axis is a monotonically increasing function of c^2, and when $c^2 \to 1$, the slope tends to $+\infty$, i.e., the major axis tends to the vertical straight line $x = \frac{1}{2}$.
- The slope of the major axis is an increasing function of k, regardless of the value of c. When $k \to 0$, the slope tends to 0 and when $k \to +\infty$, the slope tends to $+\infty$.

Appendix E: On the Relationships between μ, c, and k

Here, for each metric in Table 5, we show how to compute 1) the value of the cost reduction proportion μ that can be achieved with a technique that improves the value of a metric *PFM* from PFM_{pop} to c; 2) the improvement of *PFM* required to obtain a μ cost reduction. In Section 8.2, we showed the results only for *Precision*, which we repeat here for completeness.

For each metric *PFM*, we find the intersection point between the straight line that represent the locus in the ROC space where *PFM* is equal to a constant value c and the straight line $y = -kx + 1$ to which point $(\mu\frac{1}{1+k}, 1 - \mu\frac{k}{1+k})$ belongs. Each point on that straight line corresponds to a different value of μ and, conversely, for each value of μ that point belongs to a straight line where $PFM = c$ for a specific value of c.

Precision. We need to solve the following linear system

$$\begin{cases} y = \frac{c}{1-c}kx \\ y = -kx + 1 \end{cases}$$

The result is

$$\begin{cases} x = \frac{1-c}{k} \\ y = c \end{cases}$$

Let us equate $x = \frac{1-c}{k}$ and $x = \mu\frac{1}{1+k}$. We obtain $\mu = \frac{(1-c)(1+k)}{k}$ and $c = 1 - \frac{k\mu}{1+k}$.

Recall. The border is $y = c$, which, as we saw for *Precision* is associated with $x = \frac{1-c}{k}$. We therefore obtain exactly the same results as with *Precision*.

F-measure. We need to solve the following linear system

$$\begin{cases} y = \frac{c}{2-c}kx + \frac{c}{2-c} \\ y = -kx + 1 \end{cases}$$

Again (and as expected, since the *F-measure* is the harmonic mean of *Precision* and *Recall*), we have the same results as with *Precision* and *Recall*, since

$$\begin{cases} x = \frac{1-c}{k} \\ y = c \end{cases}$$

NPV. We need to solve the following linear system

$$\begin{cases} y = \frac{1-c}{c}k(x-1) + 1 \\ y = -kx + 1 \end{cases}$$

The result is

$$\begin{cases} x = 1 - c \\ y = -(1-c)k + 1 \end{cases}$$

By equating $x = 1 - c$ and $x = \mu\frac{1}{1+k}$, we obtain $\mu = (1-c)(1+k)$ and $c = 1 - \frac{\mu}{1+k}$.

Specificity. The border is $x = 1 - c$, which, as we saw for *NPV* is associated with $y = -(1-c)k + 1$. We therefore obtain exactly the same results as with *NPV*.

NM. We need to solve the following linear system

$$\begin{cases} y = \frac{2-c}{c}kx - 2\frac{1-c}{c}k + 1 \\ y = -kx + 1 \end{cases}$$

Again (and as expected, since *NM* is the harmonic mean of *NPV* and *Specificity*), we have the same results as with *NPV* and *Specificity*, since

$$\begin{cases} x = 1 - c \\ y = -(1-c)k + 1 \end{cases}$$

Appendix F: On the Effect of *k*

We here describe the analytic results that show the effect of k in two cases: when NC_{max} is kept fixed (Section F.1) and when μ is kept fixed (Section F.2).

F.1 Keeping NC_{max} Fixed for Different Values of *k*

Suppose that a software manager has chosen a value of NC_{max}, but he or she is not entirely sure about the value of k for the project at hand. Since the pencil of straight lines described by Formula (10) go through center point $(\mu\frac{1}{1+k}, 1 - \mu\frac{k}{1+k})$, we need to study how that point varies when NC_{max} is fixed and k varies. Depending on the possible positions of the center point, one can make decisions on the level of performance of the defect prediction models that need to be used. To this end, we solve $NC_{max} = \mu\frac{k}{(1+k)^2}$ for μ, and we obtain $\mu = NC_{max}\frac{(1+k)^2}{k}$. By substituting μ into the coordinates of center point $(\mu\frac{1}{1+k}, 1-\mu\frac{k}{1+k})$, we obtain point $(NC_{max}\frac{1+k}{k}, 1-NC_{max}(1+k))$. The x- and y-values of this point vary with k, i.e., they describe a curve in parametric form, where k is the parameter. The parametric equations of the curve are

$$x = NC_{max}\frac{1+k}{k},\ y = 1 - NC_{max}(1+k) \tag{19}$$

By solving the first for k, we obtain k as a function of x. We replace k in the second equation by using this function of x and we obtain the following relation between x and y

$$xy - (1 - NC_{max})x - NC_{max}y + NC_{max} = 0 \tag{20}$$

It can be shown that it is the equation of an equilateral hyperbola, with vertical asymptote $x = NC_{max}$ and horizontal asymptote $y = 1 - NC_{max}$. We are interested in the part of this hyperbola that is above the diagonal. It can be shown that this hyperbola intersect the diagonal at two points

$$x = \frac{1}{2} \pm \sqrt{\frac{1}{4} - NC_{max}^2} \tag{21}$$

The part under the space root sign is nonnegative, because $NC_{max} \leq \frac{1}{4}$, as we showed in Section 8.1. Thus, the software manager, based only on the desired value of NC_{max}, can compute all the points in the ROC space in which $NC = NC_{max}$.

Note also that different projects will have different values of k. If the software manager decides that all projects should have $NC = NC_{max}$, (20) also describes the set of points in the ROC space where this happens.

F.2 Keeping μ Fixed for Different Values of k

Suppose now that the software manager wants the normalized cost for every project to be at the most equal to $NC_{max} = \mu \frac{k}{(1+k)^2}$, i.e., with the same proportion μ for every value of k. The value of NC_{max} however changes across projects, but the software manager may expect that, because projects with different values of k may have different needs. We here show how center point $(\mu \frac{1}{1+k}, 1 - \mu \frac{k}{1+k})$ moves in the ROC space when k varies. Based on the definition of center point $(\mu \frac{1}{1+k}, 1 - \mu \frac{k}{1+k})$, when k vary, we have a curve in parametric form, whose parametric equations are

$$x = \mu \frac{1}{1+k}, y = 1 - \mu \frac{k}{1+k} \tag{22}$$

By eliminating k, we obtain the following relation between x and y

$$y = x + 1 - \mu \tag{23}$$

This straight line is parallel to the diagonal. The part of this straight line that belongs to the ROC space is the segment corresponding to values of x in the interval $[0, \mu]$.

References

The SEACRAFT repository of empirical software engineering data, https://zenodo.org/communities/seacraft (2017)

Alves TL, Ypma C, Visser J (2010) Deriving metric thresholds from benchmark data. In: 26Th IEEE international conference on software maintenance (ICSM 2010), september 12-18, 2010, timisoara, romania, pp 1–10. https://doi.org/10.1109/ICSM.2010.5609747

Arisholm E, Briand LC, Fuglerud M (2007) Data mining techniques for building fault-proneness models in telecom java software. In: The 18th IEEE international symposium on Software reliability, 2007. ISSRE'07. , IEEE, pp 215–224

Arisholm E, Briand LC, Johannessen EB (2010) A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. J Syst Softw 83(1):2–17. https://doi.org/10.1016/j.jss.2009.06.055

Baker SG, Pinsky PF (2001) A proposed design and analysis for comparing digital and analog mammography: Special receiver operating characteristic methods for cancer screening. Journal of the American Statistical Association 96(454):421–428. http://www.jstor.org/stable/2670280

Beecham S, Hall T, Bowes D, Gray D, Counsell S, Black S (2010) A systematic review of fault prediction approaches used in software engineering. Lero Technical Report Lero-TR-S20P1L0–2004

Beecham S, Hall T, Bowes D, Gray D, Counsell S, Black S (2010) A systematic review of fault prediction approaches used in software engineering. Tech. rep., Technical Report lero-TR-2010-04 Lero

Bowes D, Hall T, Petrić J (2018) Software defect prediction: do different classifiers find the same defects? Softw Qual J 26(2):525–552

Catal C (2012) Performance evaluation metrics for software fault prediction studies. Acta Polytechnica Hungarica 9(4):193–206

Catal C, Diri B (2009) A systematic review of software fault prediction studies. Expert Syst Appl 36(4):7346–7354

Cerpa N, Bardeen M, Kitchenham B, Verner J (2010) Evaluating logistic regression models to estimate software project outcomes. Inf Softw Technol 52(9):934–944

Cohen J (1988) Statistical power analysis for the behavioral sciences Lawrence Earlbaum Associates. routledge, New York, NY USA

da Costa DA, McIntosh S, Kulesza U, Hassan AE, Abebe SL (2018) An empirical study of the integration time of fixed issues. Empir Softw Eng 23(1):334–383. https://doi.org/10.1007/s10664-017-9520-6

Cremona L (2005) Elements of projective geometry: 3rd edn (dover phoenix editions) dover publications

Dallal JA, Morasca S (2014) Predicting object-oriented class reuse-proneness using internal quality attributes. Empir Softw Eng 19(4):775–821. https://doi.org/10.1007/s10664-012-9239-3

Dodd LE, Pepe MS (2003) Partial auc estimation and regression. Biometrics 59(3):614–623. http://www.jstor.org/stable/3695437

Drummond C, Holte RC (2006) Cost curves: an improved method for visualizing classifier performance. Mach Learn 65(1):95–130

Erni K, Lewerentz C (1996) Applying design-metrics to object-oriented frameworks. In: 3Rd IEEE international software metrics symposium (METRICS 1996), from measurement to empirical results, march 25-26, 1996, Berlin, Germany, pp 64–74. https://doi.org/10.1109/METRIC.1996.492444

Fawcett T (2006) An introduction to roc analysis. Patt Recogn Lett 27(8):861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Fenton NE, Bieman JM Software Metrics: A Rigorous and Practical Approach, Third Edition. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series. Taylor & Francis (2014). https://books.google.es/books?id=lx_OBQAAQBAJ

Flach PA (2003) The geometry of ROC space: Understanding machine learning metrics through ROC isometrics. In: Machine learning, proceedings of the twentieth international conference (ICML 2003), August 21-24, 2003, Washington, DC, USA, pp 194–201. http://www.aaai.org/Library/ICML/2003/icml03-028.php

Gini C (1912). In: Pizetti E, Salvemini T (eds) Variabilità e mutabilità. Reprinted in Memorie di metodologica statistica. Libreria Eredi Virgilio Veschi, Rome

Hall T, Beecham S, Bowes D, Gray D, Counsell S (2012) A systematic literature review on fault prediction performance in software engineering. IEEE Trans Software Eng 38(6):1276–1304

Hand DJ (2009) Measuring classifier performance: a coherent alternative to the area under the ROC curve. Mach Learn 77(1):103–123. https://doi.org/10.1007/s10994-009-5119-5

Hanley JA, McNeil BJ (1982) The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology 143(1):29–36

Hardin JW, M HJ (2002) Generalized estimating equations. CRC Press, Abingdon

Herbold S On the costs and profit of software defect prediction. Accepted for publication in IEEE Transactions on Software Engineering (01), 1–1. https://doi.org/10.1109/TSE.2019.2957794

Herbold S, Trautsch A, Grabowski J (2018) A comparative study to benchmark cross-project defect prediction approaches. IEEE Trans Software Eng 44(9):811–833. https://doi.org/10.1109/TSE.2017.2724538

Hosmer Jr, DW, Lemeshow S, Sturdivant RX (2013) Applied logistic regression. John Wiley & Sons

Huang Q, Xia X, Lo D (2019) Revisiting supervised and unsupervised models for effort-aware just-in-time defect prediction. Empir Softw Eng 24(5):2823–2862. https://doi.org/10.1007/s10664-018-9661-2

Jiang Y, Cukic B (2009) Misclassification cost-sensitive fault prediction models. In: Proceedings of the 5th international workshop on predictive models in software engineering, PROMISE 2009, Vancouver, BC, Canada, May 18-19, 2009, pp 20. https://doi.org/10.1145/1540438.1540466

Jiang Y, Cukic B, Menzies T (2008) Cost curve evaluation of fault prediction models. In: 19Th international symposium on software reliability engineering (ISSRE 2008), 11-14 november 2008, seattle/redmond, WA, USA, pp 197–206. https://doi.org/10.1109/ISSRE.2008.54

Jiang Y, Metz CE, Nishikawa RM (1996) A receiver operating characteristic partial area index for highly sensitive diagnostic tests. Radiology 201(3):745–750. https://doi.org/10.1148/radiology.201.3.8939225. PMID: 8939225

Jones C (2008) A new business model for function point metrics. http://concepts.gilb.com/dl185

Jureczko M, Madeyski L (2010) Towards identifying software project clusters with regard to defect prediction. In: Proceedings of the 6th international conference on predictive models in software engineering, pp 1–10

Kabinna S, Bezemer C, Shang W, Syer MD, Hassan AE (2018) Examining the stability of logging statements. Empir Softw Eng 23(1):290–333. https://doi.org/10.1007/s10664-017-9518-0

Khoshgoftaar TM, Allen EB (1998) Classification of fault-prone software modules: Prior probabilities, costs, and model evaluation. Empir Softw Eng 3(3):275–298. https://doi.org/10.1023/A:1009736205722

Khoshgoftaar TM, Allen EB, Jones WD, Hudepohl JP (2001) Cost-benefit analysis of software quality models. Software Quality Journal 9(1):9–30. https://doi.org/10.1023/A:1016621219262

Khoshgoftaar TM, Seliya N (2004) Comparative assessment of software quality classification techniques: An empirical case study. Empirical Software Engineering 9(3):229–257. http://www.springerlink.com/index/10.1023/B:EMSE.0000027781.18360.9b

Krantz DH, Luce RD, Suppes P (1971) Tversky a.: Foundations of measurement, vol 1. Academic Press, San Diego

Langdon WB, Dolado J, Sarro F, Harman M (2016) Exact mean absolute error of baseline predictor, MARP0. Inf Softw Technol 73:16–18

Lavazza L, Morasca S (2017) On the evaluation of effort estimation models. In: Proceedings of the 21st international conference on evaluation and assessment in software engineering, ACM, pp 41–50

Lessmann S, Baesens B, Seow HV, Thomas LC (2015) Benchmarking state-of-the-art classification algorithms for credit scoring: an update of research. Eur J Oper Res 247(1):124–136

Malhotra R (2015) A systematic review of machine learning techniques for software fault prediction. Appl Soft Comput 27:504–518

Malhotra R, Khanna M (2013) Investigation of relationship between object-oriented metrics and change proneness. Int J Mach Learn Cybern 4(4):273–286

Matthews BW (1975) Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta (BBA)-Protein Structure 405(2):442–451

McClish D (1989) Analyzing a portion of the ROC Curve. Medical decision making : an international journal of the Society for Medical Decision Making 9:190–5. https://doi.org/10.1177/0272989X8900900307

McIntosh S, Kamei Y (2018) Are fix-inducing changes a moving target? a longitudinal case study of just-in-time defect prediction. IEEE Trans Software Eng 44(5):412–428. https://doi.org/10.1109/TSE.2017.2693980

Mendling J, Sánchez-gonzález L, García F, Rosa ML (2012) Thresholds for error probability measures of business process models. Journal of Systems and Software 85(5):1188–1197. https://doi.org/10.1016/j.jss.2012.01.017

Morasca S (2009) A probability-based approach for measuring external attributes of software artifacts. In: Proceedings of the 2009 3rd international symposium on empirical software engineering and measurement, ESEM '09, Lake Buena Vista, FL, USA, October 15-16, 2009. IEEE Computer Society, Washington, pp 44–55. https://doi.org/10.1109/ESEM.2009.5316048

Morasca S (2014) Using logistic regression to estimate the number of faulty software modules. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering, EASE '14, May 13-14, 2014, ACM, New York, pp 26:1–26:9. https://doi.org/10.1145/2601248.2601297

Morasca S, Lavazza L (2017) Risk-averse slope-based thresholds: Definition and empirical evaluation. Inform Software Technol 89:37–63. https://doi.org/10.1016/j.infsof.2017.03.005

Murgia A, Ortu M, Tourani P, Adams B, Demeyer S (2018) An exploratory qualitative and quantitative analysis of emotions in issue report comments of open source systems. Empir Softw Eng 23(1):521–564. https://doi.org/10.1007/s10664-017-9526-0

Nam J, Fu W, Kim S, Menzies T, Tan L (2018) Heterogeneous defect prediction. IEEE Trans Software Eng 44(9):874–896. https://doi.org/10.1109/TSE.2017.2720603

Nucci DD, Palomba F, Rosa GD, Bavota G, Oliveto R, Lucia AD (2018) A developer centered bug prediction model. IEEE Trans Software Eng 44(1):5–24. https://doi.org/10.1109/TSE.2017.2659747

Posnett D, Filkov V, Devanbu P (2011) Ecological inference in empirical software engineering. In: Proceedings of the 2011 26th IEEE/ACM international conference on automated software engineering, IEEE Computer Society, pp 362–371

Powers DM (2011) Evaluation: from precision, recall and f-measure to roc, informedness markedness and correlation

Powers DMW (2012) The problem of area under the curve. In: 2012 IEEE International conference on information science and technology, pp 567–573. https://doi.org/10.1109/ICIST.2012.6221710

Provost FJ, Fawcett T (2001) Robust classification for imprecise environments. Mach Learn 42(3):203–231. https://doi.org/10.1023/A:1007601015854

Pundir S, Seshadri R (2012) A novel concept of partial lorenz curve and partial gini index. Int J Eng Sci Innov Technol 1(2):296–301

Radjenović D, Heričko M, Torkar R, Živkovič A (2013) Software fault prediction metrics: A systematic literature review. Inform Softw Technol 55(8):1397–1418

Ragkhitwetsagul C, Krinke J, Clark D (2018) A comparison of code similarity analysers. Empir Softw Eng 23(4):2464–2519. https://doi.org/10.1007/s10664-017-9564-7

van Rijsbergen CJ (1979) Information retrieval butterworth

Robin X, Turck N, Hainard A, Tiberti N, Lisacek F, Sanchez JC (2011) Müller, m.: proc: an open-source package for r and s+ to analyze and compare roc curves. BMC Bioinformatics 12:77. https://doi.org/10.1186/1471-2105-12-77

Sánchez-González L, García F, Ruiz F, Mendling J (2012) A study of the effectiveness of two threshold definition techniques. In: 16Th international conference on evaluation & assessment in software engineering, EASE 2012, ciudad real, spain, may 14-15, 2012. proceedings, pp 197–205. https://doi.org/10.1049/ic.2012.0026

Schneidewind NF (2001) Investigation of logistic regression as a discriminant of software quality. In: 7Th IEEE international software metrics symposium (METRICS 2001), 4-6 april 2001, london, england, pp 328–337

Shatnawi R (2010) A quantitative investigation of the acceptable risk levels of object-oriented metrics in open-source systems. IEEE Trans Software Eng 36(2):216–225. https://doi.org/10.1109/TSE.2010.9

Shatnawi R, Li W, Swain J, Newman T (2010) Finding software metrics threshold values using ROC curves. Journal of Software Maintenance 22(1):1–16. https://doi.org/10.1002/smr.404

Shepperd M, Bowes D, Hall T (2014) Researcher bias: the use of machine learning in software defect prediction. IEEE Trans Softw Eng 40(6):603–616

Shepperd M, MacDonell S (2012) Evaluating prediction systems in software project estimation. Inf Softw Technol 54(8):820–827

Singh Y, Kaur A, Malhotra R (2010) Empirical validation of object-oriented metrics for predicting fault proneness models. Software Quality Journal 18(1):3

Swets J, Pickett R (1982) Evaluation of Diagnostic Systems: Methods from Signal Detection Theory. Academic Press series in cognition and perception. Academic Press. https://books.google.it/books?id=MIx9AAAAMAAJ

Tosun A, Bener AB (2009) Reducing false alarms in software defect prediction by decision threshold optimization. In: Proceedings of the Third international symposium on empirical software engineering and measurement, ESEM 2009, October 15-16, 2009, Lake Buena Vista, Florida, USA, pp 477–480. https://doi.org/10.1145/1671248.1671311

Total Metrics (2007) Methods for software sizing – how to decide which method to use. www.totalmetrics.com/function-point-resources/downloads/R185_Why-use-Function-Points.pdf

Vilalta R, Oblinger D (2000) A quantification of distance bias between evaluation metrics in classification. In: Proceedings of the seventeenth international conference on machine learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000, pp 1087–1094

Youden WJ (1950) Index for rating diagnostic tests. Cancer 3(1):32–35

Zhang F, Keivanloo I, Zou Y (2017) Data transformation in cross-project defect prediction. Empir Softw Eng 22(6):3186–3218

Zhang H, Cheung SC (2013) A cost-effectiveness criterion for applying software defect prediction models. In: Joint meeting of the european software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, ESEC/FSE'13, saint petersburg, russian federation, august 18-26, 2013, pp 643–646. https://doi.org/10.1145/2491411.2494581

Sandro Morasca is a Professor of Computer Science at the Dipartimento di Scienze Teoriche e Applicate of the Università degli Studi dell'Insubria in Como and Varese, Italy. He was an Associate and Assistant Professor at the Politecnico di Milano in Milano and Como, Italy, and a Faculty Research Assistant and later a Visiting Scientist at the Department of Computer Science of the University of Maryland at College Park. Sandro Morasca has been actively carrying out research in Empirical Software Engineering, Software Quality, Machine Learning, Software Verification, Open Source Software, Web Services, and Specification of Concurrent and Real-time Software Systems, and has published over 30 journal papers and over 90 conference papers. Sandro Morasca has been involved in several national and international projects and has served on the Program Committees and Editorial Boards of international Software Engineering conferences and journals.

Luigi Lavazza is an Associate Professor at the University of Insubria at Varese, Italy. Formerly he was an Assistant Professor at the Politecnico di Milano, Italy. Since 1990 he has been cooperating with the Software Engineering group at CEFRIEL, where he acts as a scientific consultant in digital innovation projects. His research interests include: Empirical software engineering, software metrics and software quality evaluation; Software project management and effort estimation; Software process modeling, measurement and improvement; Open Source Software. He was involved in several international research projects, and he also served as reviewer of EU projects. He is co-author of over 170 scientific articles, published in international journals, in the proceedings of international conferences, or in books. He has served on the PC of several international Software Engineering conferences, and in the editorial board of international journals.

Affiliations

Sandro Morasca[1] · **Luigi Lavazza**[1]

Luigi Lavazza
luigi.lavazza@uninsubria.it

[1] Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria, Padiglione Rossi, Via Ottorino Rossi 9, I-21100, Varese, Italy