A proposed method to evaluate and compare fault predictions across studies

Barbara Russo
Free University of Bozen-Bolzano
Faculty of Computer Science
P.za domenicani, 3
Bozen, Italy
Barbara.Russo@unibz.it

ABSTRACT

Studies on fault prediction often pay little attention to empirical rigor and presentation. Researchers might not have full command over the statistical method they use, full understanding of the data they have, or tend not to report key details about their work. What does it happen when we want to compare such studies for building a theory on fault prediction? There are two issues that if not addressed, we believe, prevent building such theory. The first concerns how to compare and report prediction performance across studies on different data sets. The second regards fitting performance of prediction models. Studies tend not to control and report the performance of predictors on historical data underestimating the risk that good predictors may poorly perform on past data. The degree of both fitting and prediction performance determines the risk managers are requested to take when they use such predictors. In this work, we propose a framework to compare studies on categorical fault prediction that aims at addressing the two issues. We propose three algorithms that automate our framework. We finally review baseline studies on fault prediction to discuss the application of the framework.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*Product metrics, performance measures*

General Terms

Measurement, Performance, Reliability, Theory

Keywords

fault, confusion matrix, machine learning, model comparison

1. INTRODUCTION

The majority of the studies on software fault prediction do not provide value to the empirical research as they should, [1]. In these studies, researchers are often not enough skilled to use complex statistical models and tune their parameters appropriately or they do not have a full command over the intrinsic limits of their data sets. Such studies do not help building a proper theory on fault prediction. Hall et al, [1] ascribe the cause to researchers reporting insufficient contextual and methodological details and neglecting the influence that context and statistical techniques have on model performance. In their literature review, the authors find that simple models that do not need to tune parameters typically perform better. When models require parameter tuning, researchers often opt to use default values or automatic tools with which they have little control on the tuning process. Researchers might additionally overestimate the intrinsic limits of the data sets they collected. For example, models performance is not guaranteed when categories of model output (e.g., faulty or non-faulty) are imbalanced. As reported by the authors, none of the many studies analysed investigated the problem. As consequence, the way researchers understand tools and data affect the way they report their work and the development of a data science on fault prediction. This can be a real problem in particular with the rapid increase of data complexity and size, [2].

In this work, we discuss two issues that, in our opinion, prevent research on fault prediction to develop any comparative analysis across studies: 1) measuring and reporting the ability of a model to fit the actual system before using it for prediction on new fresh data and 2) understanding the dimensionality of the prediction problem and use the measures of performance accordingly.

Nowadays, research on big data puts great emphasis on the prediction power of models [2] over their sole explanatory ability: a model must both represent the current and the future behavior. This is not that easy to achieve, though. The point is that models that have good prediction ability might perform poorly in fitting historical data and viceversa. This situation can be unavoidable as it might be simply due to the population characteristics on which data mining is performed. In any case, for decision making, measuring and reporting fitting and prediction ability of models is of paramount importance. For example, managers must be informed about the risk of using excellent prediction models with poor fitting ability, as such models have a complexity that does represent the actual system behavior (model under-fitting or over-fitting, [3]). We will see that the majority of the baseline studies in Hall et al [1] do not report fitting accuracy.

Studies on fault prediction report model performance with different measures. This might make a comparison harder. In particular, to compare two or more studies, we need to consider two facts. First, when the a priori and a posteriori classifications are given, fault prediction performance is a two dimensional problem. The a priori classification is the distribution of faulty and non-faulty instances in a data set. The distribution characterizes the population from which the data set is taken and is typically measured by the percentage of faulty instances. The a posteriori classification is determined by a threshold against which the model probability is evaluated. Typically this threshold is set to 0.5. Thus, an instance is predicted as faulty if the probability of the model on the instance is greater than 0.5, as non-faulty otherwise. The area under the curve of the Receiving Operator Characteristic measures independency from the threshold value. Two dimensional problem means that 1. when a priori and a posteriori classifications are given, any additional performance measure can be derived from two independent ones with appropriate formulas (e.g., Conversion Formula for precision, [4]), 2. one single performance measure is not enough to compare models built with different methods (e.g., misclassification rate) or overlook some predictive behavior of the classification model (e.g., expected cost of misclassification), [5]. Therefore, it is not important if researchers use different measures in their studies. What matters is use at least two independent measures, the nature of the data, and the probability threshold set for the models to be used.

Studies on fault prediction often do not describe the nature of the data used. Such habit can prevent building a comparative theory across studies and limit the results of any literature synthesis on the topic. We will review the baseline papers selected by Hall et al. [1] to exemplify the problem.

In this study, we first formalise the theory of fault prediction and provide a simple map for managers to summarize and visualize performance of fault predictors for a better comparison. We also present three algorithms that serve to perform, report, and compare studies on fault prediction. Finally, we review those studies that adhere to the criteria of good empirical research on fault prediction identified by Hall et al. [1] to understand to what extend they can be used to build a theory.

In Section 2, we formulate our research hypothesis. We introduce the classification problem and the measures of performance in Section 3 respectively where we also discuss the dimensionality of the classification problem. In Section 3, we present and reformulate the classification problem presenting the double nature as data characteristics and model quality. The algorithms and the risk map are presented in Section 4. We review the baseline studies in Section 5. We conclude in Section 6

2. RESEARCH HYPOTHESIS

Synthesis of primary studies has the goal to understand the state of the art and the open problems that a specific research needs to solve in its process of knowledge building. Mining repositories to learn models on fault prediction is a very active field and provides secondary research with a generous set of studies. Recent findings of secondary research are not enthusiastic of the scientific rigor that such primary studies show [1]. By far, this is due to the fact that software engineers must learn to be data scientists. Data science requires a strong command over statistical instruments and understanding of data intrinsic limits, [2]. In addition, to build a science, researchers need to be able to compare studies and comparison requires primary studies reporting their findings with sufficient rigor.

We believe that even synthesis of research on fault prediction does not have full command over the generalization instruments that enable researchers to build solid theories from evidence. In this paper, we will highlight two major issues that affect both primary and secondary existing studies.

3. THE CLASSIFICATION PROBLEM

In this section, we describe the fault prediction problem as presented in Munson and Khoshgoftaar, 1992 [6] for classification of software modules by changes. The same approach has been recently used to classify faulty software modules in Nagappan et al., [7–9]. This approach focuses on dichotomous output categories. This is the typical setting in research on fault or error prediction. Per this strategy, there is an *a priori classification* of modules into two mutually exclusive groups. A criterion variable is used for the group assignment. For example, a module is classified with a code of zero if it has been found faulty, or with a code of 1 otherwise. The a priori classification is determined by the total number of instances in a set (e.g., the number of modules in a system) and the percentage of instances in one of the two groups (e.g., the proportion of faulty modules). A classifier computes the *posterior classification* of group membership associating to an input a given probability to belong to a group. Depending on a pre-set decision rule (e.g., probability greater than 0.5) an input is classified in one or in the other group. For instance, [6], and [8], use the logistic probability of module attributes grater than 0.5 to define predicted fault modules. The module is then assigned to the group for which it has the greatest probability of membership. False and true positive rates, precision and true positive rate are then calculated to evaluate the performance of the posterior probabilities against the a priori classification.

3.1 The confusion matrix

Assume that the a priori classification divides data into two groups G_1 (faulty modules) and G_2 (not faulty modules). An output is Positive if it is in G_1, it is Negative if it is in G_2. True Positives (TP) or True Negatives (TN) are, respectively, the number of actual fault or actual not fault instances that have been correctly classified. False Positives (FP) or False Negative (FN) are respectively the number of actual not fault or actual fault instances that have been misclassified. The confusion matrix, Table 1, describes the performance of the classification problem in terms of TP, TN, FP, and FN for given a priori and a posteriori classifications.

3.2 Composite measures of performance

Measures of classification performance are calculated from the confusion matrix that defines the classification problem. In Table 2, we report the ones most relevant to this study. These measures depend on both the model and the a priori classification defined for a given population. In particular, they depend on the margins of the classification matrix, Pos and Neg, determined by the a priori classification Thus, to compare models with these measures on different data sets,

Table 1: Confusion matrix

	Pred. Pos	Pred. Neg
Pos	TP	FN
Neg	FP	TN

the sets must have the same a priori classification.

Table 2: Measures of model performance, [10], [11], [1]

Name	Formula
Misclassification rate, (MR)	$\dfrac{FP+FN}{FP+FN+TP+TN}$
True Positive rate, recall (TP_r)	$\dfrac{TP}{FN+TP}$
False Positive rate, (FP_r)	$\dfrac{FP}{FP+TN}$
Balance	$1-\sqrt{\dfrac{(TP_r-1)^2+(FP_r)^2}{2}}$
Precision	$\dfrac{TP}{FP+TP}$

Conversion Formula, [4]

Precision$=\dfrac{1}{1+\frac{FP_r}{TP_r}\cdot\frac{1}{r}}$,

Neg = TN+FP

Pos = TP+FN

r$=\dfrac{Pos}{Neg}=\dfrac{Pos\%}{(1-Pos\%)}$

3.3 The dimension of the prediction problem

Once the a posteriori classification is known, a classification problem on data sets with the same a priori classification can be solved with only two independent performance measures, as they completely determine the confusion matrix. For example, Table 3 shows this for TP_r, FP_r.

It would be more practical to compare models by one single measure of performance. Unfortunately, recent research reports that one single performance measure is not enough, [5], [11]. For example, with the misclassification rate

$$MR=\frac{FP+FN}{Pos+Neg}.$$

we cannot decide between two models when one has simultaneously greater FP and lower FN than the other. Existing studies use different measures. Hall et al. [1] report that there is no common agreement on which measures to use.

Table 3: Confusion matrix in terms of TP_r, FP_r, Pos, and total size n.

	Pred. Pos	Pred. Neg
Pos	TP_r*Pos	(1-TP_r)*Pos
Neg	FP_r*(n-Pos)	(1-FP_r)*(n-Pos)

For example, Gray et al. [12] advocate the use of precision and TP_r, especially in case of imbalanced sets (i.e., sets with r far from 1) as "Relying on TP_r and FP_r or methods based around them, including: ROC analysis, AUC-ROC, and the balance metric, can present an overly optimistic view of an algorithm's performance if there is a large skew in the class distribution. Precision is required to give a more accurate representation of true performance in this context." We believe that the point is actually not related to the choice of two specific measures as we can always transform two performance measures in other two when we know the a priori classification, [1], [13], [14]. The point is that the classification problem is also determined by the a priori classification and any comparison among different data sets must take this into account. For example, the conversion formula in Table 2 defines a hyperbolic relation between precision and the ratio $\frac{FP_r}{TP_r}$ that varies according to the a priori classification parameter r, Fig. 1. For every percentage p, the conversion formula implies that

$$\frac{FP_r}{TP_r}\leq r*(\frac{p}{1-p}) \text{ iff precision} \geq p \qquad (1)$$

(iff holds for equality too). In Fig. 1 we set p=50% and

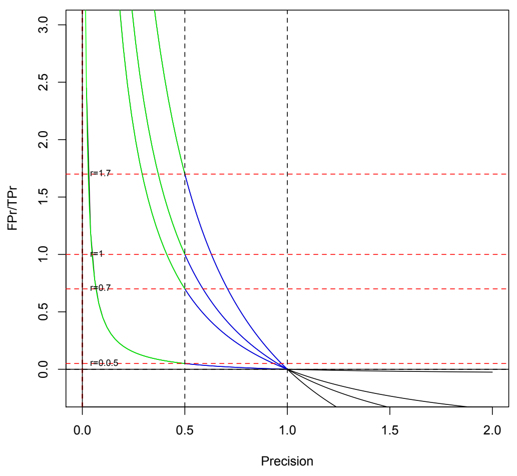

Figure 1: $\frac{FP_r}{TP_r}$ against precision. The curve varies with the values of the a priori classification parameter r. In the figure, p=50%

draw different hyperboles according to the different values of r. The blue segment of these curves represents all the pairs

(FP_r, and TP_r), that have $FP_r \cdot \frac{1}{r} \leq TP_r$ or equivalently precision greater than 50%. We can see that decreasing r, the length of the blue curve segment as well as the chance of getting models with high precision decreases. For example, models with true positive rate 70% and false positive rate 3% would never have precision greater than 50% on data sets with Pos%≤4% (i.e., r≤ 1/24). In the ROC plan defined by FP_r and TP_r, the points on the blue curve fall above the line $y = \frac{1}{r}x$. Fig. 2 plots the ROC plan in the case r < 1, e.g., the typical inequality in fault prediction (and imbalanced data has r much less than 1, [15]). Varying r varies the slope of the line and the areas above and below the line where precision is respectively greater and lower 50%. In a more general case for any given value of p, the two areas are delimited by a line with slope

$$(1/r) * (p/(1-p)) \tag{2}$$

and the area above or below the line has precision greater or lower than p respectively.

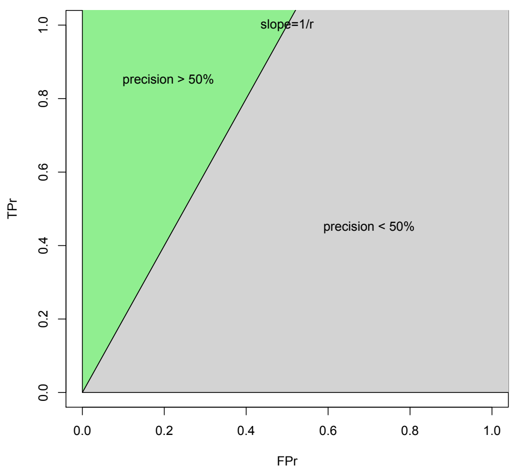

Figure 2: Areas of high and low precision with respect to the a priori classification parameter r. In the figure, r< 1 and p=50%.

Our formalisation of the classification problem can also contribute to a recent debate on the relevance of precision in assessing performance of fault predictors. According to Zhang and Zhang [4], when data is imbalanced, models can detect a good number of faults (TP_r is high) and raise few false alarms (FP_r is small), but still be not precise in prediction (precision is small). To prove this fact, they introduce the conversion formula in Table 2. For this reason, the authors claim that true and false positive rate are not sufficient to determine the full model performance in case of imbalanced data and precision must be computed. On the other side, imbalanced data sets are very frequent in empirical software engineering and precision is an unstable measure. Menzies et al. [11] show that precision computed for different types of learning machines and applied on resample data sets shows the highest variation among performance measures. With our formalisation, it results more evident that the solution to the debate resides on variation of the parameter r. With the conversion formula, now we know that the variation of the precision found in Menzies et al. [11] could be ascribed to the variation of r across the resampled data sets as precision depends on r and FP_r/TP_r and that TP_r and FP_r can be used when one can control the variation of the parameter r. Controlling for this variation can be done with technique of data balancing, [15], but balancing is not always feasible and requires high Pos, [16].

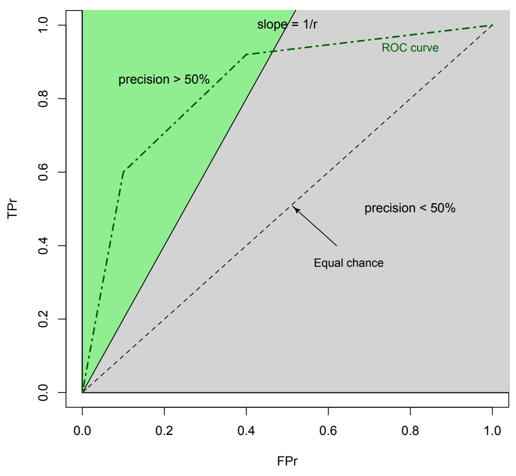

Figure 3: ROC curve

Until here we have illustrated the classification problem in case the a posteriori classification is given. The ROC curve, the segmented line in Fig. 3, pictures the relation between TP_r, FP_r, at different a posteriori classification. In the figure, we use two thresholds that correspond to two a posteriori classifications. The Area Under the Curve (AUC) of the ROC curve is a measure of model performance independent from the threshold of the a posteriori classification. AUC estimates the probability that a random selected positive instance gets higher probability to be classified positive than a randomly selected negative instance to be classified negative. The diagonal plots the point in which the probability is even. As any single composite measure, AUC can-

```
X // data set
p // positive float number lees than 1
/*compute a priori classification on X */
compute r
M // classification model
/*Set the decision rule*/
for instance $\in$ X
   if P(instance) $\geq p$
      instance =Pos
   else
      instance =Neg
/*compute measure of performance according to the decision rule*/
compute $TP_r$
compute $FP_r$
return r, $TP_r$, $FP_r$
```

Figure 4: The classification algorithm.

not be used alone to compare models across different studies even when the a priori classification is the same, [17]. Fig. 4 summarises the final classification algorithm that includes the definition of both the a priori and the a posteriori classification.

3.4 Comparing models on different data sets

To obtain the ROC curve, the algorithm is repeated for different p. To compare a model on two or more data sets with different value of r, we apply the algorithm two or more times, build the areas of precisions each time, and finally overlap them. In case of two models, the results is shown in Fig. 5. In the figure, only models with TP_r and FP_r in the top left area guarantee precision greater than 50% in the two data sets. Good models across data sets need to fall in the intersection of all top left triangles possibly approaching the value (0,1).

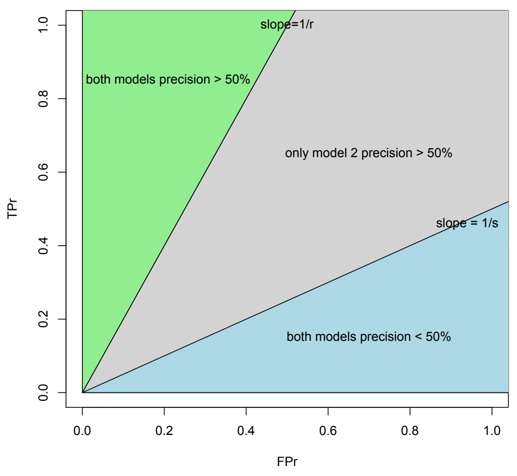

Figure 5: Areas of high and low precision with respect to two a priori classifications (r, s). In the figure $r < 1 < s$.

4. THE TTV ALGORITHM: TRAIN, TEST, AND VALIDATE

We introduce here the TTV algorithm (Fig. 6) that computes the measures of fitting and prediction performance that we propose in this study. The algorithm has three stages in which it: 1. trains models on data, 2. tests models on new fresh data, and 3. validates models on further new data for prediction. The TTV algorithm aims at formalizing the process so that all stages are included and for each stage the appropriate result is output for recoding.

To train models the algorithm uses cross-validation. Cross-validation is a process that randomly splits the original data set into k-subsets of equal size (typically, k=10) and trains a model on the union of k-1 subsets and test it on the remaining one. In literature, cross-validation is typical used to train and validate data with the consequence that testing is skipped. Results on the training performance are not reported too. TTV uses cross-validation to determine the best parameters of a model. For example, in neural networks training is used to derive the best topology of the network.

The algorithm uses new fresh data to test the trained model and computes the model performance on this data. Testing on new data provides performance measures that are independent from the data used to build the model. Performance measures on testing data indicate the accuracy of the model to render the actual system behavior.

```
X // original data set
/* Split X into train, test, and validation sets */
$X_V$ = 30% * X // Validation set
C=X-$X_V$
$X_{Tr}$ = 60% * C  // Training set
$X_{Ts}$ = 40% * C // Test set
/* Cross-validation on $X_{Tr}$ to tune parameters */
for i in 1:10
    /* Randomly split $X_{Tr}$ in 10 equal size subsets */
    $Y_i \subset X_{Tr}$ and $|Y_i| = \frac{|X_{Tr}|}{i}$ and $Y_i \cap Y_j =$ for $i \neq j$
    return $I=\{Y_i\}$
M($\tau$) // Model with parameters $\tau$
for i in 1:10
    /* aggregate 9 subsets to train the model M($\tau$) */
    $W_i = \cup_{j \neq i} Y_j$
    train on $W_i$
    /* Test on the remaining one */
    test on $Y_i$
    /* select the best parameters
    according to the best statistic of reference */
    compute $s_i$ on $Y_i$
    $i_0$ :  $s_{i_0} =$ best $s_i$
    return $\tau_{i_0}$
M = M($\tau_{i_0}$)
/* Evaluate accuracy in fitting */
run classification algorithm for M on $X_{Ts}$
/* Evaluate performance in prediction */
run classification algorithm for M on $X_V$
```

Figure 6: The TTV algorithm

Finally, the TTV algorithm uses further new fresh data to make predictions. Measures of performance on this data represent the future reliability of the system assuming that the system behaves in the future as it does currently.

4.1 Accuracy of fit and prediction performance

Fig. 7 is what we call the risk map. The map is a semantic representation of the ROC plan. It summarises the prediction problem in a graphical representation that can be better presented to the manager. The risk map combines the performance of the model in fitting (accuracy of fitting) with the performance of the same model in prediction. The upper triangle represent the area in which models have precision greater than 50% on both test 9performance in fitting) and validation sets (performance in prediction). Models must aim at falling in this area.

4.2 Interpreting confusion matrices

We assume that we ran the TTV algorithm and we obtained values of r, TP_r, and FP_r for both test and validation sets. Now, we are able to construct the confusion matrix for both sets. As a matter of example, we use values from a real case study, whose data we can access to, [16]. The results on testing and validation and the values for the a priori

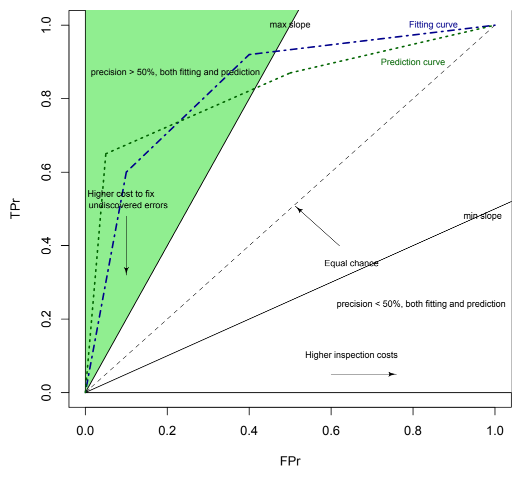

Figure 7: The risk map

classification are:

	TP_r	FP_r	Pos%	size
Test set	86%	5%	39%	21
Valid. set	82%	11%	49%	35
Original set			40%	106

Tables 4 and 5 illustrate the two corresponding confusion matrices. In the following, we interpret the two confusion matrices of Table 4 and 5. To our knowledge the only other study reporting the two matrices in fault prediction is in Khoshgoftaar et al., [15]. In this case, the authors claimed that the two measures were enough similar to represent system reliability in both fitting and prediction. We argue that the degree of similarity must be spelled out. To this end, we will use the risk map.

Table 4: Confusion matrix on test set, derived from Pedrycz et al. [16]

	Pred. Pos	Pred. Neg	Total
Pos	7 86%	1 14%	8 100%
Neg	1 5%	12 95%	13 100%
Total Percent	8 38%	13 62%	21 100%

We use the confusion matrix on the test set to describe the actual system behavior. The Pos percentage indicates that out of any 1000 instances of the population representing the system 390 are faulty. The multilayer perceptron is able to capture about 335 faulty instances (86%*390) with precision of 91% rising 30 false alarms (5%*610) and 54 faults left undiscovered (14%*390). Based on the current experience managers can set a weight for the cost that incurs for inspecting instances that are not faulty and unexpected cost of maintenance due to undiscovered faulty instance. In this example we set the same value for the two types of costs.

Table 5: Confusion matrix on validation set, derived from Pedrycz et al. [16]

	Pred. Pos	Pred. Neg	Total
Pos	14 82%	3 18%	17 100%
Neg	2 11%	16 89%	18 100%
Total Percent	16 45%	19 54%	35 100%

To predict the future behavior of the system we use the confusion matrix derived on the validation set. We first assume that the behavior of the application in the next three months is similar to the current one. Thus, we assume that the expected Pos% is the one we found on the test set, i.e., 39%. Then we suppose that 1000 sequence abstractions are being isolated. Therefore, we expect to have about 610 non-faulty and about 390 faulty instances. We apply the multilayer perceptron and predict those features. Per Table 5, we would expect the model to identify about 450 faulty instances (45% x 1000). Developers could inspect and test these types with more accuracy. We would expect they waste time on 67 instances that are actually not faulty (11% x 610). We would predict the model to identify correctly about 320 instances that are actually faulty (82% x 390), but fail to identify 70 faulty instances (18% x 390) that will cause future cost of inspection. Comparing the test and the validation results, we see that in future costs will increase. With which accuracy managers need to take this costs into consideration? If we use the risk map, we simply have to plot the two pair of values of TP_r and FP_r. the maximal slope is 1.56 and corresponds to the a priori distribution of the test set. We can easily see that the two pairs of values are in the upper triangle of the risk map defined for r=1.56 and p=50%. Precision is then greater than 50%. With the map the manager can in parallel evaluate the risk that the model does not fit the actual system behavior and the risk that the model predicts wrong instances assuming that the model fits 100% the actual behavior. In literature, the second type of risk is measured by the misclassification rate MR. In the example above, MR is 8% in fitting and 14% in prediction. As we mentioned, MR is not enough to compare the quality of two models, though, as it is not able to distinguish between two other types of risk: i) the risk of wasting time in fixing faults where there are none (values with high FP_r in Fig 3) ii) the risk of not inspecting instances when they should, which causes unbudgeted costs of maintenance (values with low TP_r in Fig 3).

4.3 Comparing predictions

Fig. 7 is the map a manager is presented with. Each model obtained with the TTV algorithm is mapped into two points in the plan defined by FP_r and TP_r. The two points represent the model fitting and prediction performance of the model according to TP_r and FP_r. The minimal value of the ratio r computed on test and validation sets determine the upper triangle and the maximal slope of the border line. If the two points are above the line with maximal slope, by eq. 0 we know that the model has precision greater than a value p (in the figure 50%). Using the distance from the (0,1) point, we can additionally identify a new area in the risk map. It is a disc with center ($FP_r = 0$,$TP_r = 1$) and ray R. Points falling in the disc are "enough" near to the absolute best performance result (0,1). To determine a value of R that guarantees proximity with the point (0,1) and precision greater that p, we compute the point in which the circle of the disc is tangent to the border line of the upper triangle. Solving a simple a algebraic problem, we find that:

$$R_0 = \frac{r_{min} * (1-p)}{\sqrt{p^2 + r_{min}^2 * (1-p)^2}} \tag{3}$$

and consequently

$$balance_0 = 1 - \frac{r_{min} * (1-p)}{\sqrt{2} * \sqrt{p^2 + r_{min}^2 * (1-p)^2}} \tag{4}$$

If we assume p$\geq$ 50% and r$\leq$ 1 (i.e., imbalanced data sets as in typical fault prediction studies), from eq. 1, we get the slope of the border line greater than 1. As such the disc of ray R_0 will always fall above the diagonal of the risk map (i.e. equal chance line). Thus, the prediction problem reduces to find those models whose TP_r and FP_r values determine a point in the disc of ray R_0 for both test and validation sets. When models satisfy this minimal requirement, they can be compared with other models or models on different data sets. The final comparison algorithm is based on this rule. In Fig. 8, we illustrate the case of two models and two dat sets but the algorithm can be easily extended to multiple models and data sets.

```
X // data set
Y // data set
p // positive float number lees than 1
M // classification model
N // classification model
run the TTV algorithm for M on X
$P^X_{test}$ =($FP_r$,$TP_r$) on test set of X
$P^X_{val}$ =($FP_r$,$TP_r$) on validation set of X
compute $r_X = \min\{r_{test}, r_{val}\}$ for X
compute $R_0$ with $r_{min} = r_X$
if dist($P^X_{test}$, $(0,1)$) < $R_0$ & dist($P^X_{val}$, $(0,1)$) < $R_0$
   $r_X$, $P^X_{val}$
else
   $x = 0$
run the TTV algorithm for N on Y
$P^Y_{test}$ =($FP_r$,$TP_r$) on test set of Y
$P^Y_{val}$ =($FP_r$,$TP_r$) on validation set of Y
compute $r_Y = \min\{r_{test}, r_{val}\}$ for Y
compute $R_0$ with $r_{min} = r_Y$
if dist($P^Y_{test}$, $(0,1)$) < $R_0$ & dist($P^Y_{val}$, $(0,1)$) < $R_0$
   $r_Y$, $P^Y_{val}$
else
   $r_Y = 0$
if ($r_X = 0$ & $r_Y = 0$)
    $r_0 = 0$
else
   compute $r_0 = \min\{r_X, r_Y\}$
/* compare M and N on X and Y*/
compute $R_0$ with $r_{min} = r_0$
if (dist($P^X_{val}(N)$, $(0,1)$) < $R_0$ & dist($P^Y_{val}(M)$, $(0,1)$) < $R_0$)
   if ($P^X_{val}(N)$, $(0,1)$) <dist($P^Y_{val}(M)$, $(0,1)$)
      return $P^X_{val}$
   else
      return $P^Y_{val}$
else
   if (dist($P^X_{val}(N)$, $(0,1)$) < $R_0$)
      return $P^X_{val}$
   else if dist($P^Y_{val}(M)$, $(0,1)$) < $R_0$)
     return $P^Y_{val}$
   else
     return $\Phi$
```

Figure 8: The comparison algorithm

5. RE-REVIEWING STUDIES IN MINING DATA FOR FAULT PREDICTION

We analyze the twenty-one categorical papers used in Hall et al [1] that report sufficient contextual and methodological details to be synthesized. These papers provide categorical models classifying instances by their faultiness (dichotomous classification). The authors compare such papers by AUC, F-measure, true positive rate, and precision. The F-measure can be written in terms of true positive rate and precision. As such, the papers are ultimately compared by AUC, precision and true positive rate. We see indeed that not all the papers report AUC and the majority sets to 0.5 the threshold for the a posteriori classification.

We reviewed all the papers according whether the study reported:

1. The a priori classification (the r value)
2. Measures that could be transformed into TP_r and FP_r
3. Accuracy of fitting
4. Any cross-validation technique and if this technique used the training, testing, and validation procedure

Table 6 summarizes our findings. In the table, we illustrate whether the a priori classification is specified only for original, or also for training, testing, or validation sets. The comparison algorithm works the best with all these details. We also reported all the measures of model performance that we found in the papers and not only the one used in the Hall et al. paper. This is because we need to understand whether a paper solve completely the classification problem. In other words, we check whether the measures can determine the confusion matrix and therefore can be transformed into TP_r and FP_r. We also check whether fitting performance is reported. We found that in the majority of the study it is not reported. This is also due to the fact that many researchers use the n-fold cross validation as a black box statistical instrument that outputs only performance on the kth fold used for prediction. In this way, researchers do not have any command on the accuracy of the model on the remaining n-1 folds.

Finally, only 4 studies more or less satisfy our criteria that are conducted two different group of researchers (Simula lab and at the Empirical Software Engineering Laboratory at Florida Atlantic University). Arisholm et al. [18] use the TVV approach splitting the original data set into three disjoint sets. The a priori classification is reported only for the training set which is then balanced (r=1) for the training. Precision and TP$_r$ are reported for the test and validation sets, but with no information on the a priori classification of these sets we are not able to derive the FP$_r$ and the confusion matrix. Arisholm et al. [19] apply the same approach that in their previous paper [18], but this time they use also FP$_r$. Unfortunately, this time the result of the fitting and prediction performance are not reported separately and again we cannot map their result onto the risk map. What is missing in Khoshgoftaar et al. [5] are only the measures useful to fully derive the confusion matrix or equivalently the pair TP$_r$ and FP$_r$. The paper that provides all details to run our algorithm is the one of Khoshgoftaar and Seliya [20]. Fig. 9 illustrates the authors' result mapped in the risk map. The figure reports the border lines for the test set (release 2) used for fitting accuracy and the validation set (release 4) for prediction performance. The lines are reported for both p=10% (solid lines) and p=50% (dashed lines). The points are the values of FP$_r$ and TP$_r$ for seven different models for release 2 (red) and 4 (black). From the larger circle the manager can see that some models have precision less than 10% in prediction and all the models have precision for both fitting and prediction less than 50% (smaller circle). This result is not much satisfactory.

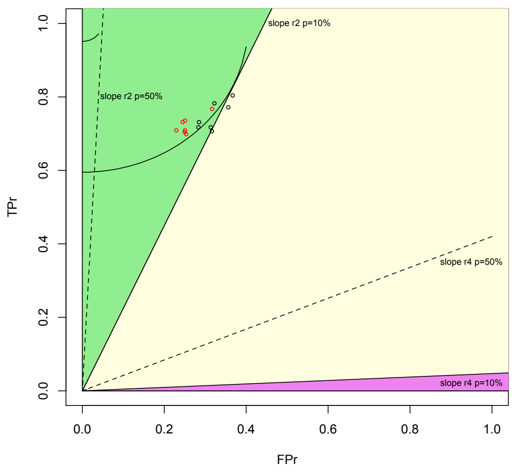

Figure 9: The risk map for the study of Khoshgoftaar and Seliya [20]

6. CONCLUSIONS

In this study, we propose a framework to conduct fault prediction analyses. The framework is expressed as three consecutive algorithms and want to control two crucial issues that can hamper a comparison across studies: compute and report accuracy of fitting of a model and determine the set of parameters and variables needed to describe the solution and the context of a classification problem. Overall, the present study aims at building the needed environment to generalize results across studies on fault predictions. Such environment is unfortunately still missing, [1]. We also reviewed the baseline studies that result compliant with the quality criteria of Hall et al. [1]. Among the twenty-one studies we found four that have almost all the characteristics of rigor to apply our framework. We selected one and applied framework and produced the risk map. The risk map partition the ROC plan in areas related to the values of precision. In this way managers can better evaluate the risk to select models according to TP$_r$, FP$_r$ and the nature of the data.

7. REFERENCES

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

[2] V. Dhar, "Data science and prediction," *Commun. ACM*, vol. 56, no. 12, pp. 64–73, Dec. 2013.

[3] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. The MIT Press, 2010.

[4] H. Zhang and X. Zhang, "Comments on "data mining static code attributes to learn defect predictors"," *Software Engineering, IEEE Transactions on*, vol. 33, no. 9, pp. 635–637, 2007.

[5] T. Khoshgoftaar, K. Gao, and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction," in *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, vol. 1, Oct 2010, pp. 137–144.

[6] J. Munson and T. Khoshgoftaar, "The detection of fault-prone programs," *Software Engineering, IEEE Transactions on*, vol. 18, no. 5, pp. 423–433, 1992.

[7] N. Nagappan, A. Zeller, T. Zimmermann, K. Herzig, and B. Murphy, "Change bursts as defect predictors," in *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on*, Nov 2010, pp. 309–318.

[8] N. Nagappan, B. Murphy, and V. Basili, "The influence of organizational structure on software quality: an empirical case study," in *Proceedings of the 30th international conference on Software engineering*, ser. ICSE '08. New York, NY, USA: ACM, 2008, pp. 521–530.

[9] N. Nagappan and T. Ball, "Use of relative code churn measures to predict system defect density," in *Proceedings of the 27th international conference on Software engineering*, ser. ICSE '05. New York, NY, USA: ACM, 2005, pp. 284–292.

[10] C. M. Bishop, *Neural Networks for Pattern Recognition*, 1st ed. Oxford University Press, USA, Jan. 1996.

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

[12] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "Further thoughts on precision," in *Evaluation Assessment in Software Engineering (EASE 2011), 15th Annual Conference on*, April 2011, pp. 129–133.

Table 6: Review of the baseline papers. #code refers to the numbering used in Hall et al. [1]

Study	Report a priori classification (r)?	Measures	Report fitting performance?	Use TTV?
#8 Arisholm et al. 2007 [18]	Partially. Reported only for training set that has been artificially balanced (r=1)	AUC, cost effectiveness, F-measure, precision, TP_r	Yes	Yes
#9 Arisholm et al. 2010 [19]	Partially. Reported only for training set that has been artificially balanced (r=1)	AUC, cost effectiveness, F-measure, FP_r, FN_r, precision, TN_r, TP_r	Yes	Yes
#18 Bird et al. 2009 [21]	No	AUC, F-measure, precision, TP_r	No	No. Use 50-times random split, 2/3-1/3 and across versions validation
#21Caglayan et al. 2009 [22]	Partially. Reported only for the original set	FP_r, TP_r	No	No. Use cross validation and across versions validation
#37 Denaro & Pezzé 2002 [23]	Yes. Reported only for the original set but cross-validation is stratified	Accuracy, precision, TP_r	No	No. Stratified cross-validation, i.e. partitions have a priori distribution similar to the original data set
#51 Gylmothy et al. 2005 [24]	Partially. Reported only for the original set	Defect detection rate, F-measure, precision, TP_r	No	No. Use 10-fold cross validation
#56 Zhang 2009 [25]	Partially. Reported only for the original set	F-measure, Precision, TP_r	No	No. Use 10-fold cross validation
#74 Kaur & Malhotra 2008 [26]	Partially. External reference. Only for the original set	Accuracy, AUC, F-measure, precision, TP_r	No	No. Use k-fold cross validation
#76 Khoshgoftaar et al., 2010 [5]	Yes. Reported for test and validation sets.	AUC	Yes	Yes. Traditional and balancing technique
#83 Khoshgoftaar & Seliya, 2004 [20]	Yes. Reported for training and validation sets	Expected Cost of Misclassification, FP_r, FN_r	Yes	Yes. Use across versions TTV but the splitting is not completely disjoint
#86 Khoshgoftaar et al. 2002 [27]	Partially. Reported only for training set	FN_r, FP_r	No	No. Use across releases validation
#109 Mende & Koschke 2010 [28]	Partially. Reported only for the original sets	Cost effectiveness, defect detection rate, precision, TP_r,	No	No. Ten times 10-fold cross valiadation
#110 Mende et al. 2009 [29]	Partially. Reported only for the original sets	AUC, Precision, defect detection rate, deviation from optimal model, TP_r,	No	No. n-fold cross validation
#116 Mizuno et al. 2007 [30]	Partially. Reported only for the original sets that has also been artificially balanced (r=1)	Accuracy, confusion matrix, precision, TP_r	No	No. 10-fold cross validation
#117 Mizuno & Kikuno 2007 [31]	No.	Accuracy, confusion matrix, FP_r, FN_r, precision, TP_r	No	No. Only classification
#118 Moser et al. 2008 [32]	Partially. Reported only for the original set	Precision, TP_r, FP_r	No	No. Use 10-fold cross validation
#120 Nagappan et al. 2010 [7]	Yes.	Precision, TP_r	No	No. Use 50-times random split, 2/3-1/3
#154 Schroeter et al. 2006 [33]	Partially. Reported only for the original sets	Precision, TP_r	No	No. Use 40-times random split, 2/3-1/3
#160 Shatnawi & Li [34]	Partially. Reported only for the original sets.	AUC	No	No. Use across versions validation
#164 Shivaji et al. 2009 [35]	Partially. Reported only for the original sets.	Accuracy, AUC, precision, TP_r	No	No. Classification only
#203 Zhou et al. 2010 [36]	Partially. Reported only for the original sets.	AUC	No	No. Use 10-fold cross validation and across versions validation

[13] D. Bowes, T. Hall, and D. Gray, "Dconfusion: a technique to allow cross study performance evaluation of fault prediction studies," *Automated Software Engineering*, vol. 21, no. 2, pp. 287–313, 2014.

[14] ——, "Comparing the performance of fault prediction models which report multiple performance measures: Recomputing the confusion matrix," in *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, ser. PROMISE '12. New York, NY, USA: ACM, 2012, pp. 109–118. [Online]. Available: http://doi.acm.org/10.1145/2365324.2365338

[15] T. M. Khoshgoftaar, E. B. Allen, J. P. Hudepohl, and S. J. Aud, "Application of neural networks to software quality modeling of a very large telecommunications system," *Trans. Neur. Netw.*, vol. 8, no. 4, pp. 902–909, jul 1997.

[16] W. Pedrycz, B. Russo, and G. Succi, "Mining system logs to learn error predictors: A case study of a telemetry system," *Journal of Empirical Software Engineering*, p. to appear, 2014. [Online]. Available: https://pro.unibz.it/staff/brusso/Publications/TOC_Revised_New.pdf

[17] M. Shepperd, D. Bowes, and T. Hall, "Researcher bias: The use of machine learning in software defect prediction," *Software Engineering, IEEE Transactions on*, vol. 40, no. 6, pp. 603–616, June 2014.

[18] E. Arisholm, L. Briand, and M. Fuglerud, "Data mining techniques for building fault-proneness models in telecom java software," in *Software Reliability, 2007. ISSRE '07. The 18th IEEE International Symposium on*, Nov 2007, pp. 215–224.

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

[20] T. Khoshgoftaar and N. Seliya, "Comparative assessment of software quality classification techniques: An empirical case study," *Empirical Software Engineering*, vol. 9, no. 3, pp. 229–257, 2004.

[21] C. Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu, "Putting it all together: Using socio-technical networks to predict failures," in *Software Reliability Engineering, 2009. ISSRE '09. 20th International Symposium on*, Nov 2009, pp. 109–119.

[22] B. Caglayan, A. Bener, and S. Koch, "Merits of using repository metrics in defect prediction for open source projects," in *Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009. FLOSS '09. ICSE Workshop on*, May 2009, pp. 31–36.

[23] G. Denaro and M. Pezzè, "An empirical evaluation of fault-proneness models," in *Proceedings of the 24th International Conference on Software Engineering*, ser. ICSE '02. New York, NY, USA: ACM, 2002, pp. 241–251.

[24] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *Software Engineering, IEEE Transactions on*, vol. 31, no. 10, pp. 897–910, Oct 2005.

[25] H. Zhang, "An investigation of the relationships between lines of code and defects," in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, Sept 2009, pp. 274–283.

[26] A. Kaur and R. Malhotra, "Application of random forest in predicting fault-prone classes," in *Advanced Computer Theory and Engineering, 2008. ICACTE '08. International Conference on*, Dec 2008, pp. 37–43.

[27] T. M. Khoshgoftaar, X. Yuan, E. B. Allen, W. D. Jones, and J. P. Hudepohl, "Uncertain classification of fault-prone software modules," *Empirical Softw. Engg.*, vol. 7, no. 4, pp. 297–318, Dec. 2002.

[28] T. Mende and R. Koschke, "Effort-aware defect prediction models," in *Software Maintenance and Reengineering (CSMR), 2010 14th European Conference on*, March 2010, pp. 107–116.

[29] T. Mende, R. Koschke, and M. Leszak, "Evaluating defect prediction models for a large evolving software system," in *Software Maintenance and Reengineering, 2009. CSMR '09. 13th European Conference on*, March 2009, pp. 247–250.

[30] O. Mizuno, S. Ikami, S. Nakaichi, and T. Kikuno, "Spam filter based approach for finding fault-prone software modules," in *Mining Software Repositories, 2007. ICSE Workshops MSR '07. Fourth International Workshop on*, May 2007, pp. 4–4.

[31] O. Mizuno and T. Kikuno, "Training on errors experiment to detect fault-prone software modules by spam filter," in *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ser. ESEC-FSE '07. New York, NY, USA: ACM, 2007, pp. 405–414.

[32] R. Moser, W. Pedrycz, and G. Succi, "A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," in *Proceedings of the 30th international conference on Software engineering*, ser. ICSE '08. New York, NY, USA: ACM, 2008, pp. 181–190.

[33] A. Schröter, T. Zimmermann, and A. Zeller, "Predicting component failures at design time," in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, ser. ISESE '06. New York, NY, USA: ACM, 2006, pp. 18–27.

[34] R. Shatnawi and W. Li, "The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process," *J. Syst. Softw.*, vol. 81, no. 11, pp. 1868–1882, Nov. 2008.

[35] S. Shivaji, E. J. W. Jr., R. Akella, and S. Kim, "Reducing features to improve bug prediction," in *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 600–604.

[36] Y. Zhou, B. Xu, and H. Leung, "On the ability of complexity metrics to predict fault-prone classes in object-oriented systems," *J. Syst. Softw.*, vol. 83, no. 4, pp. 660–674, Apr. 2010.