Comments

Comments on "Data Mining Static Code Attributes to Learn Defect Predictors"

Hongyu Zhang, *Member*, *IEEE*, and Xiuzhen Zhang

Abstract—In this correspondence, we point out a discrepancy in a recent paper, "Data Mining Static Code Attributes to Learn Defect Predictors," that was published in this journal. Because of the small percentage of defective modules, using Probability of Detection (pd) and Probability of False Alarm (pf) as accuracy measures may lead to impractical prediction models.

Index Terms—Defect prediction, accuracy measures, static code attributes, empirical.

1 Introduction

In the January 2007 issue of this journal, a paper titled "Data Mining Static Code Attributes to Learn Defect Predictors" [1] was published. In that paper, Probability of Detection (pd) and Probability of False Alarm (pf) are used to measure the accuracy of a defect prediction model. Their models generate average results of $pd = 71\%$ and $pf = 25\%$. The authors of [1] consider these results satisfactory and draw their conclusions based on them. This correspondence points out the limitation of using pd and pf as accuracy measures in imbalanced classification. Using the Recall/Precision measures, we show that the models built in [1] are not satisfactory for practical use and should be improved.

2 The Evaluation of Defect Prediction Models

Prediction of defective modules can be cast as a classification problem in machine learning: Given training samples of modules with labels as defective (Positive) or nondefective (Negative), a classification model can be learned from the training data. The model is then used to classify unknown modules. A prediction model has four results: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), as shown in Table 1. The total number of actual defective modules is denoted POS and the total number of actual nondefective modules is denoted NEG.

To evaluate the accuracy of predictions, [1] uses the receiver operating characteristic (ROC) curves, which consist of pd and pf. A single measure $balance$ is also used to balance between pd and pf. These measures are defined as follows:

$$pd = \frac{TP}{TP+FN},\ pf = \frac{FP}{FP+TN},$$
$$balance = 1 - \frac{\sqrt{(0-pf)^2 + (1-pd)^2}}{\sqrt{2}}. \tag{1}$$

- H. Zhang is with the School of Software, Tsinghua University, Beijing 100084, China. E-mail: hongyu@tsinghua.edu.cn.
- X. Zhang is with the School of Computer Science and Information Technology, RMIT University, Melbourne 3001, Australia. E-mail: zhang@cs.rmit.edu.au.

Manuscript received 15 Jan. 2007; accepted 4 June 2007; published online 13 June 2007.
Recommended for acceptance by M. Harman.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0302-1206.
Digital Object Identifier no. 10.1109/TSE.2007.70706.

For evaluating the performance of a prediction model, another set of accuracy measures is Recall and Precision, which is widely used in the used in the Information Retrieval area [2], [3]:

$$\mathrm{Rec}all = \frac{TP}{TP+FN},\ \mathrm{Pr}ecision = \frac{TP}{TP+FP},$$
$$F\text{-measure} = \frac{2 \times \mathrm{Rec}all \times \mathrm{Pr}ecision}{\mathrm{Rec}all + \mathrm{Pr}ecision}. \tag{2}$$

Recall is actually the same as pd, which defines the ratio of detected true defective modules in comparison to the total number of defective modules. Precision defines the ratio of correctly detected modules. A good prediction model should achieve high Recall and high Precision. A single measure, the F-measure, is used to combine Recall and Precision. It is defined as the harmonic mean of Precision and Recall (a more general form of F-measure is defined as a weighted harmonic mean of Precision and Recall). The values of Recall, Precision, and F-measure are between 0 and 1; the higher, the better.

Based on (1) and (2), we know that

$$\mathrm{Pr}ecision = \frac{TP}{TP+FP} = \frac{1}{1+\frac{FP}{TP}} = \frac{1}{1+\frac{NEG \times pf}{POS \times pd}}. \tag{3}$$

Using the data given in Fig. 3 and Fig. 12 of [1], we calculate the Precision values based on (3). The results are shown in Table 2 (the values in *italic* are the original data from [1]).

We notice that the Precision and F-measure values are very low for all data sets (except KC4). For example, for the CM1 data set, the Precision is 20.64 percent, which means that, if a module is predicted as defective, the probability of it actually being defective is only 20.64 percent. For the PC2 data set, if the prediction model claims that a module is defective, the probability of it actually being defective is only 2.02 percent. These results are considered unsatisfactory, although the pd values are high. Therefore, defect prediction through such models would not be very useful in practice. Applying such models would defeat the very purpose of defect prediction, which is about allocation of limited QA resources more efficiently (so that efforts can be concentrated on the potentially problematic modules).

The models with high pd and low pf do not necessarily lead to accurate models with high precision. The reason is that the distribution of classes (defective or nondefective) is highly imbalanced. The number of nondefective modules is much greater than the number of defective modules. As shown in Table 2, the percentage of defective modules in each data set (except for the KC4 data set) is very low (ranging from 0.4 percent to 12 percent). From (3), we can see that the Precision could be low if the *NEG/POS* ratio is high. The only exception, the KC4 data set, has the *NEG/POS* ratio 1.04; therefore, a high pd/pf ratio leads to a high Precision. For all other data sets, the number of nondefective modules are 7-249 times more than the defective modules; therefore, their Precision is low even though their pd is high and pf is low.

TABLE 1
The Results of a Prediction Model

		Predicted		
		Defective	Non-defective	
Actual	Defective	TP	FN	POS (TP+FN)
	Non-defective	FP	TN	NEG (FP+TN)

TABLE 2
The Prediction Results

Data Set	# modules	% defective	$\frac{NEG}{POS}$	*pd* (%)	*pf* (%)	balance	Recall (%)	Precision (%)	F-measure	Expected *pf* (for Precision = 60%)
CM1	*506*	*9*	10.11	*71*	*27*	0.72	71	**20.64**	0.32	4.68
KC3	*459*	*9*	10.11	*69*	*28*	0.70	69	**19.60**	0.31	4.55
KC4	*126*	*49*	1.04	*79*	*32*	0.73	79	**70.34**	0.74	50.60
MW1	*404*	*7*	13.29	*52*	*15*	0.64	52	**20.69**	0.30	2.61
PC1	*1108*	*6*	15.67	*48*	*17*	0.61	48	**15.27**	0.23	2.04
PC2	*5590*	*0.4*	249.0	*72*	*14*	0.78	72	**2.02**	0.04	0.19
PC3	*1564*	*10*	9.00	*80*	*35*	0.71	80	**20.25**	0.32	5.93
PC4	*1458*	*12*	7.33	*98*	*29*	0.79	98	**31.55**	0.48	8.91

Table 2 also gives the expected *pf* values for each data set if the Precision reaches 60 percent. We can see that, in order to achieve Precision of 60 percent, the original *pf* values shown in Fig. 12 of [1] should be further improved.

To verify our results, we have also repeated the study described in [1], using the same NASA data sets, the naive Bayes (with log-transforms) learner, and the WEKA tool. The results confirm that the prediction models proposed in [1] are impractical for software defect prediction due to the low precisions.

3 Conclusion

In this comment, we have shown that the models built in [1] are not satisfactory for practical use. We suggest using Recall/Precision, instead of *pd*/*pf*, to measure the accuracy of a software defect prediction model.

References

[1] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," *IEEE Trans. Software Eng.*, vol. 33, no. 1, Jan. 2007.
[2] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval.* ACM Press, 1999.
[3] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.