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

Hongyu Zhang
School of Software
Tsinghua University
Beijing 100084
China
hongyu@tsinghua.edu.cn

Xiuzhen Zhang
School of Computer Science and
Information Technology
RMIT University, Melbourne 3001
Australia
zhang@cs.rmit.edu.au

Abstract

In this correspondence, we point out some discrepancies 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 *pd* and *pf* as accuracy measures may lead to impractical prediction models.

Keywords: defect prediction, accuracy measures, static code attributes, empirical

1. Introduction

In the January 2007 issue of this journal, a paper entitled "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 non-defective (Negative),

a classification model can be learnt 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 as POS and the total number of actual non-defective modules is denoted as NEG.

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

Table 1: The results of a prediction model

To evaluate the accuracy of a prediction, [1] uses Receiver-Operator (ROC) curves, which consists of Probability of Detection (*pd*) and Probability of False Alarm (*pf*). A single measure *balance* is also defined to balance between *pd* and *pf*. These measures are defined as follows:

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

For evaluating performance of a prediction model, another set of accuracy measures is Recall and Precision, which is widely used in Information Retrieval area.

$$\mathrm{Re}call = \frac{TP}{TP + FN},\ \mathrm{Pr}ecision = \frac{TP}{TP + FP}, F - measure = \frac{2 \times \mathrm{Re}call \times \mathrm{Pr}ecision}{\mathrm{Re}call + \mathrm{Pr}ecision} \qquad (2)$$

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. The 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. The values of Recall, Precision and F-measure are between 0 and 1, the higher the better.

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

$$\Pr ecision = \frac{TP}{TP + FP} = \frac{1}{1 + \frac{FP}{TP}} = \frac{1}{1 + \frac{NEG \times PF}{POS \times PD}} \qquad (3)$$

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

Data Set	**# modules**	**% defective**	**$\frac{\text{NEG}}{\text{POS}}$**	**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. The Prediction Results

We notice that the Precision and F-measure values are very low for all datasets (except the KC4). For example, for the CM1 dataset, the Precision is 20.64%, which means that if a module is predicted as defective, the probability of it actually being defective is only 20.64%. For the PC2 dataset, if the prediction model claims a module defective, the probability of it actually being defective is only 2.02%. 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 non-defective) is highly imbalanced. The number of non-defective modules is much more than the number of defective modules. As shown in Table 2, the percentage of defective modules in each dataset (except the KC4 dataset) is very low (ranging from 0.4% to 12%). From the Equation (3) we can see that the Precision could be low if the *NEG/POS* ratio is high. The only exception KC4 dataset has the *NEG/POS* ratio 1.04 therefore high *pd/pf* ration leads to high Precision. For all other datasets, the number of non-defective modules are 7-249 times more than the defective modules, therefore their Precision is low even their *pd* is high and *pf* is low.

Table 2 also gives the expected *pf* values for each dataset if the Precision reaches 60%. We can see that in order to achieve Precision 60%, the original *pf* values shown in Figure 12 of [1] shall be further improved.

To verify our results, we have also repeated the study described in [1], using the same NASA datasets, 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 correspondence, 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. 32, no. 11, Jan 2007.