# Problems with Precision

Tim Menzies, *Member, IEEE,* Alex Dekhtyar, Justin Distefano, and Jeremy Greenwald

## I. Introduction

Zhang&Zhang [15] (hereafter, the Zhangs) argue that the low precision detectors seen in Menzies, Greenwald, and Frank's paper *Data Mining Static Code Attributes to Learn Defect Predictors* [13] (hereafter, DMP) are "not satisfactory for practical purposes". They demand that "a good prediction model should achieve both high Recall and high Precision" (which we will denote as "high precision&recall"). All other detectors, they argue, "may lead to impractical prediction models".

We have a different view and this short note explains why. While we disagree with the Zhangs' conclusions, we find that their derived equation is an important result. The insightful feature of the *Zhangs' equation* is that it can use information about the problem at hand to characterize the pre-conditions for high precision and high recall detectors. To the best of our knowledge, no such characterization has been previously reported (at least, not in the software engineering literature).

## II. Precision Instability

*Precision instability* is the real reason that we do not assess performance in terms of precision. But precision instability was not discussed in the DMP paper. Hence, the Zhangs' are right to complain about our selection of assessment criteria.

We first detected precision instability in several NASA data sets. If a researcher wants to demonstrate that detector generator $A$ is better than method $B$, then they must measure the performance of $A$ and $B$, under a variety of treatments. Figure 1 shows one such study [14] where different learners (e.g. M5', J48, ROCKY, LSR) were:

- *Trained* on one of five data sets;
- *Self-applied* on the same training set (to generate a baseline performance measure);
- Then *tested* on each of the other four data sets.

The results are shown in Figure 1. On most performance measures, the detectors were remarkably stable and similar performances were observed in different data sets. The exception was $prec$ (precision) which had very large standard deviations when applied to different data sets. Hence, if detectors were to be assessed in terms of precision, it would be very difficult to show that method $A$ was better than $B$.

Another problem with precision instability is maintainability. It is important to stabilize a project's defect detectors so that they remain viable after release. For example, no project manager wants to discover that their 80% precise detector is only 25% precise when a project update is pushed out 3 months later.

We show below that the Zhangs' equation can explain precision instability, as well as certain other prior empirical results.

tim@menzies.us (LCSEE, WVU)
dekhtyar@cs.uky.edu (CS, UK)
jdistefano@ismwv.com (Integrated Software Metrics)
jgreen@cs.pdx.edu (CS, PDX)
See http://menzies.us/pdf/07precision.pdf for an earlier draft of this paper. Manuscript received Feburary 21, 2007; revised March 11, 2007.

## III. The Mathematics of Precision

Let $\{A, B, C, D\}$ denote the true negatives, false negatives, false positives, and true positives (respectively) found by a binary detector. Certain standard measures can be computed from $A, B, C, D$:

$$\begin{array}{ll} pd = recall = & \frac{D}{B+D} \\ pf = & \frac{C}{A+C} \\ prec = precision = & \frac{D}{D+C} \\ acc = accuracy = & \frac{A+D}{A+B+C+D} \\ selectivity = & \frac{C+D}{A+B+C+D} \\ neg/pos = & \frac{A+C}{B+D} \end{array}$$

The last measure ($neg/pos$) is most important to the subsequent discussion. The Zhangs' equation is derived as follows:

$$prec = \frac{D}{D+C} = \frac{1}{1+\frac{C}{D}} = \frac{1}{1+neg/pos \cdot pf/recall} \quad (1)$$

which can be rearranged to

$$pf = \frac{pos}{neg} \cdot \frac{(1-prec)}{prec} \cdot recall \quad (2)$$

Note that, in Equation 2, when $recall$ is fixed then false alarm rate becomes controlled by precision and a fixed constant determined by the data set being examined; i.e. when ($\alpha = neg/pos$) and $recall = 1$ then:

$$pf = \alpha \cdot \frac{1-prec}{prec} \quad (3)$$

From Equation 3, it is clear that for any targeted recall value, *increasing* precision requires *decreasing* false alarm rates; e.g. for $prec \in \{0.5, 0.70, 0.9, 0.95\}$, $pf$ becomes $\{1, 0.43, 0.11, 0.005\}$, respectively. The effect is particularly marked for data sets with large $neg/pos$ ratios (e.g. like the data processed by DMP).

## IV. Large $Neg/Pos$ Ratios

A detail not explored by the Zhangs is that many software engineering data sets have extremely large $neg/pos$ ratios. For example:

- In the DMP paper, the data sets studied had $neg/pos$ ratios of 1.04, 7.33, 9, 10.11, 13.29, 15.67, and 249.
- Hayes, Dekhtyar and Sundaram [9] use text mining to find pairs of connected requirements in a corpus of 220 requirements and 235 design elements (the same CM-1 dataset used in [13]). The total number of possible links in the dataset is $220 \cdot 235 = 51,700$, while the the ground truth RTM contains 361 links, for the $neg/pos$ ratio of $51,700/361 = 143.2$.
- For an extreme example, Google reports that over $10^9$ web pages contain the phrase "software" but only one them is the home page of this journal. Hence, $neg/pos$ for web searching is at least $10^9$.

Figure 2 graphs Equation 1 for the DMP $neg/pos$ ratios. Figure 3 does the same, but is restricted to zones of higher precision: only the surface for $0.5 \leq prec \leq 1$ is shown. That shadow of the surface on the bottom plane shows that this zone of high precision, high recall, and large $neg/pos$. As $neg/pos$ increases, high recall&precision is only possible when $pf$ becomes vanishingly small. For example, in the ranges $0.65 \leq prec, recall \leq 0.8$, Equation 2 reports that $pf$ falls into the following ranges:

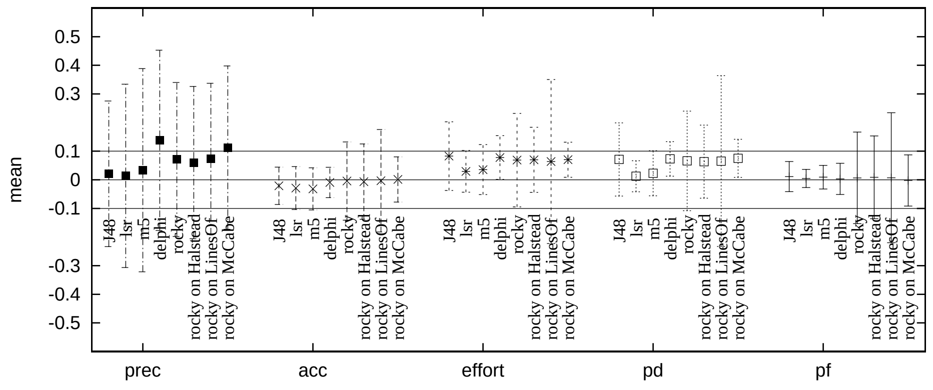

Fig. 1. Mean $\mu$ and standard deviation $\sigma$ of changes in defect detector statistics between a baseline (tested on the training set) and another data set (tested on different data). A zero value denotes that the detector worked the same on training and test data. Dots denote mean ($\mu$) values. Whiskers extend from $\mu+\sigma$ to $\mu-\sigma$. The data sets used in this study had some overlap with the DMP data; i.e. $cm1, kc1, kc2, jm1, pc$. From [14].

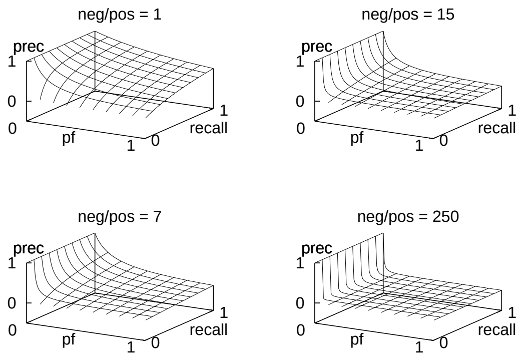

Fig. 2. The relationship between $pd, prec, recall$ and $neg/pos$.

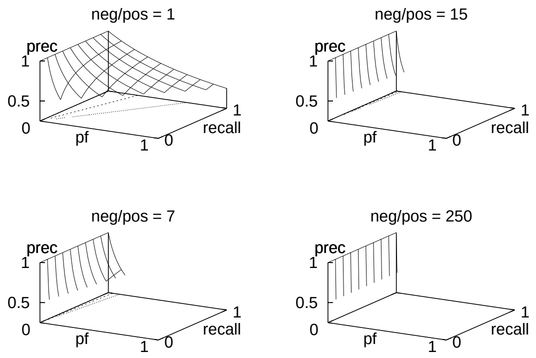

Fig. 3. Figure 2, cropped to the region where $prec > 0.5$.

- $0.023 \le pf \le 0.062$ for $neg/pos = 7$;
- $0.0108 \le pf \le 0.0287$ for $neg/pos = 15$;
- $0.007 \le pf \le 0.0017$ for $neg/pos = 250$;

Detectors learned in the domain of software engineering rarely yield high precision detectors (see Figure 4). Using the Zhangs' equations, the reasons for this are very clear:

- Those detectors all try to maximize recall;
- Figure 2 shows that such detectors can only achieve high precision in the rare case of very low $pf$.

Not only do the Zhangs' equation explain the Figure 4 results, they also inform the the instability of precision *and* the stability of $pf$ and $pd$ (recall) seen in Figure 1:

- Note how, in Figure 2, that at very small $pf$ values, tiny changes in $pf$ can lead to very large changes in $prec$ (sudden jumps from zero to one).
- The other measures in Figure 2, on the other hand, change far more smoothly and slowly.

That is, the Zhangs' equation is the essential theoretical statement needed to explain numerous prior results such as those shown in Figure 4 (i.e. [1], [2], [4], [9], [12], [13], [14]).

## V. When Low Precision is Useful

Achieving both high precision& recall can be problematic. As shown by Zhangs' equation, optimizing for one often compromises the other (especially for data sets with large $neg/pos$ ratios). Fortunately, there are may industrial situations where low precision and

- The DMP paper achieved recalls over 70% and minimum false alarm rates of 15%. Using the Zhangs' equation and data from DMP, it can be shown that the DMP precisions were quite low: $\{min, median, max\} = \{2\%20\%, 70\%\}$ (the last number came from a data set with $neg/pos \approx 1$).
- Huang et.al. [4] won "best paper" at the 2006 IEEE Requirements Engineering conference with detectors exhibiting $prec \approx 0.25$.
- Without extensive feedback from human experts, the classifiers used by Hayes, Dekhtyar, and Sundaram exhibited $pf > 0.6$ (hence, very low precisions) [9].
- Other such as Antoniol et.al. [1], [2] and Marcus et.al. [12] researchers into software tracability report that high recall is achievable only with low precision detectors. Antoniol has also been exploring bug traces in Mozilla components and has found the same high-precision, low-recall trade-off.
- After much experiments with linear regression, model tree learners, Bayes classifiers, decision tree learners, singleton rule learners, and some home-brew learners [14], the general trend is very clear. For those learners, executing on the DMP datasets, $pf \le pd - 0.5$; That is, for those learners and those data sets, obtaining high recall figures of $pd > 0.6$ implies $pf > 0.1$ and, consequently, low precision.

Fig. 4. Some low precisions seen in the SE literature.

high recall detectors are useful. For example, one of us (DiStefano) has used our low-precision detectors to review flight code developed at the NASA Glenn Research Center (Ohio, USA). When the results of these detectors were presented to the lead flight engineer, he confirmed that the identified sections (which did not have any recorded defects) had been problematic to maintain, and contained several bugs which had not yet been entered into the defect system.

For another example, from outside the field of SE, a user of commercial web search engine like Google can quickly flick through (say) three pages of results before finding a page of interest. Google has so many return customers since even with precisions of (say) $\frac{1}{30}$, the effort involved in looking at a page is so low that users don't mind examining 29 false alarms.

More generally, there are several situations where low precision detectors are useful:

- *When the cost of missing the target is prohibitively expensive.* In mission critical or security applications, the goal of 100% recall may be demanded in all situations, regardless of the precision.
- *When only a small fraction the data is returned.* Hayes, Dekhytar, & Sundaram call this fraction $selectivity$ and offer an extensive discussion of the merits of this measure [9].
- *When there is little or no cost in checking false alarms.* For example, a detector we have found useful in industrial settings is to check modules where

  $$\frac{lines\ of\ comments}{lines\ of\ code} > 0.25$$

  This detector triggers on complex functions that programmers comment extensively, instead of splitting up into smaller, more maintainable, functions. This detector is imprecise- it often triggers on well-written functions with detailed comments. However, based on commercial experience, we assert that it is fast and simple for a human agent to inspect the identified modules and discern which ones were well-written and which were over-commented to disguise being badly coded. We use this detector to find code that should be rewritten prior to release.

## VI. Research Directions

Just because high precision&recall detectors have not been seen before in SE does not mean that this goal is impossible. If large $neg/pos$ ratios are the problem, then perhaps the solution is to change those ratios in the training data. In "oversampling", the minority class ($pos$) is repeated multiple times. In "under-sampling", some portion of the majority class ($neg$) is discarded. In this way, a training data set with $neg/pos = 1$ might be generated. Figure 2 shows that the space of $neg/pos = 1$ detectors contains many candidates with both high precision&recall. However, while a promising technique, there is contradictory evidence for its value. Yiu reports that the appropriate re-sampling technique and appropriate classifier is dataset dependent [3]. In limited studies with one learner and a few data sets, Drummond & Holte [6] found that oversampling had little value. They offer some evidence for the value of under-sampling but concluded that other methods can do better. One issue is that while re-sampling yields training set contains $neg/pos = 1$, the test set still has the original distributions. In any case, this area is ripe for further exploration.

Another promising direction might be to try boosting. The AdaBoost algorithm [7] builds an ensemble of detectors 1,2,3,... where detector $i$ is built from problems that were misclassified by detectors $1, 2, .., i - 1$. AdaBoost defines a voting procedure for making conclusions after passing all new test instances to every member of the ensemble. It can be shown that increasing the size of the ensemble decreases error [7]. That is, the goal of both high precision&recall might be achievable using boosting. Curiously, to do so, we require using detectors that generates enough false alarms to inform the boosting.

The theoretical advantage of boosting has yet to show significant improvements in real-world defect data sets (see the modest improvements of [10] or the poor comparative performance of AdaBoost compared to other methods in [11, p64-77]). Therefore, to address the Zhangs' challenge of high precision&recall, we need to look for other techniques.

Yet another avenue to explore is *stacking*; i.e. levering the strengths and weaknesses of different learners in an assembly that does better than any single learner. For example, Gaddam et.al. achieved high precision&recall in one data set by combining clustering with decision tree learning. However, stacking is a poorly understood area and the behavior of the resulting assembly is difficult to predict. Gaddam et.al.'s toolkit only reached high precision&recall in one dataset; in several others, it could not [8].

Our final suggestion for how to achieve the Zhangs' goal is to augment automatic learning with some user modeling. Starting with unsupervised learners, it is possible to give learners information about the top candidate conclusions. A feedback loop can then be entered as a kind of as iterative supervised learning (results from generation $i$ inform and improve the results at generation $i+1$). This methodology leads to two different things you can measure: (1) the recall/precision of the *current* output, or (2) the recall/precision of the list of candidate conclusions used for learning *during multiple iterations*:

- In [9], the former is measured and Hayes, Dekhytar, & Sundaram iteratively refine their learned detectors, transforming low precision detectors into high precision detectors. In one case study, the reached precisions and recalls increased over 0.85 (in the $current$ output) after five rounds of users reviewing and commenting on the learned detectors. The challenge with this method is that is requires extensive involvement by knowledgeable users- and such users can be a scarce and expensive resource. Researchers exploring this approach must balance the benefits of high precision&recall detectors against their high construction cost.
- In subsequent work [5], Hayes and Dekhytar measured the recall/precision of the candidates used *during multiple iterations*. They found that they were still faced with the same low precision-high recall trade-off; e.g., to have seen 90% of the correct candidates, we had to go through a list whose precision is about 20%.

In summary, while the above techniques show promise, it may take much further research to achieve high precision&recall detectors in software engineering data sets with large $neg/pos$ ratios.

## VII. Conclusion

The Zhangs argue that predictors are useless *unless* they have high precison&recall. We have a different view, for two reasons. Firstly, for SE data sets with large $neg/pos$ ratios, it is often required to lower precision to achieve higher recall. Secondly, there are many domains where low precision detectors are useful.

Nevertheless, there is much value in Zhangs' equation. It is a useful result that explains numerous prior results such as [1], [2], [4], [9], [12], [13], [14]. The Zhangs' equation also explains why precision is much less stable than other measures. Hence, researchers are advised not to use precision when assessing their detectors. Other measures are more stable (i.e. recall ($pd$) and false alarm rates), especially for data sets with large $neg/pos$ ratios.

## References

[1] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo. Recovering traceability links between code and documentation. *IEEE Transactions on Software Engineering*, 28(10):970–983, October 2002.

[2] G. Antoniol and Yann-Gael Gueheneuc. Feature identification: A novel approach and a case study. In *ICSM 2005*, pages 357–366, 2005.

[3] Alexander Yun chung Liu. The effect of oversampling and undersampling on classifying imbalanced text datasets. Master's thesis, 2004. Available from `http://www.lans.ece.utexas.edu/~aliu/papers/aliu_masters_thesis.pdf`.

[4] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc. The detection and classification of non-functional requirements with application to early aspects. In *RE 2006*, pages 36–45, 2006.

[5] A. Dekhtyar, J.H. Hayes, and J. Larsen. Make the most of your time: How should the analyst work with automated traceability tools? In *3rd International Workshop on Predictive Modeling in Software Engineering (PROMISE'2007)*, 2007.

[6] C. Drummond and R. C. Holte. C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on Learning from Imbalanced Datasets II*, 2003.

[7] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *JCSS: Journal of Computer and System Sciences*, 55, 1997.

[8] S.R. Gaddam, V.V. Phoha, and K.S. Balagani. K-means+id3: A novel method for supervised anomaly detection by cascading k-means clustering and id3 decision tree learning methods. *IEEE Transactions on Knowledge and Data Engineering*, 19(3), March 2007.

[9] Jane Huffman Hayes, Alex Dekhtyar, and Senthil Karthikeyan Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *IEEE Trans. Software Eng*, 32(1):4–19, 2006.

[10] T.M. Khoshgoftaar, E. Geleyn, L. Nguyen, and L. Bullard. Cost-sensitive boosting in software quality modeling. In *IEEE Symposium on High Assurance Software Engineering*, volume 00, page 51, Los Alamitos, CA, USA, 2002. IEEE Computer Society.

[11] Y. Ma. *An Empirical Investigation of Tree Ensembles in Biometrics and Bioinformatics*. PhD thesis, January 2007.

[12] A. Marcus and J. Maletic. Recovering documentation-to-source code traceability links using latent semantic indexing. In *Proceedings of the Twenty-Fifth International Conference on Software Engineering*, 2003.

[13] Tim Menzies, Jeremy Greenwald, and Art Frank. Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, January 2007. Available from `http://menzies.us/pdf/06learnPredict.pdf`.

[14] Tim Menzies and Justin S. Di Stefano. How good is your blind spot sampling policy? In *2004 IEEE Conference on High Assurance Software Engineering*, 2003. Available from `http://menzies.us/pdf/03blind.pdf`.

[15] H. Zhang and X. Zhangu. Comments on 'data mining static code attributes to learn defect predictors'. *IEEE Transactions on Software Engineering*, September 2007.