Further Thoughts on Precision

David Gray, David Bowes, Neil Davey, Yi Sun and Bruce Christianson

Abstract—
Background: There has been much discussion amongst automated software defect prediction researchers regarding use of the *precision* and *false positive rate* classifier performance metrics.
Aim: To demonstrate and explain why failing to report *precision* when using data with highly imbalanced class distributions may provide an overly optimistic view of classifier performance.
Method: Well documented examples of how dependent class distribution affects the suitability of performance measures.
Conclusions: When using data where the minority class represents less than around 5 to 10 percent of data points in total, failing to report *precision* may be a critical mistake. Furthermore, deriving the *precision* values omitted from studies can reveal valuable insight into true classifier performance.

I. Introduction

It is surprisingly difficult to characterise appropriately the performance of data mining classification algorithms from the field of machine learning. Deciding which performance measures to use involves taking several factors into account, including the costs associated with misclassification, and the class distribution. We believe the inappropriate use of classifier performance measures to be a current problem in the reporting of defect prediction research, and this short paper explains why.

In January 2007 the Menzies et al. paper *'Data Mining Static Code Attributes to Learn Defect Predictors'* was published in the IEEE Transactions on Software Engineering (Menzies, Greenwald & Frank 2007). In this study many defect prediction experiments were carried out. Classifier performance was reported using the two metrics used in *receiver operating characteristic (ROC)* analysis, the *true positive rate* and the *false positive rate*. In the journal these metrics were referred to as the *probability of detection (pd)* and the *probability of false alarm (pf)*, respectively. The use of these metrics motivated a comments paper by Zhang and Zhang (Zhang & Zhang 2007). They argued that the prediction "models built in (Menzies, Greenwald & Frank 2007) are not satisfactory for practical use". This was because the *precision;* the proportion of modules predicted as being defective which were also originally labeled as being defective, was low for 7 out of the 8 data sets (between 2.02 and 31.55 percent). The authors conclude by suggesting that, for reporting on the performance of software defect prediction models, the true positive rate be used with precision rather than with the false positive rate.

All authors are with the Computer Science Department at the University of Hertfordshire, UK. Respective email addresses are: {d.gray, d.h.bowes, n.davey, y.2.sun, b.christianson}@herts.ac.uk

The Zhang and Zhang comments paper motivated a response by two of the original journal authors and two others (Menzies, Dekhtyar, Distefano & Greenwald 2007). Here the main arguments were that "detectors learned in the domain of software engineering rarely yield high precision", and that low precision predictors can be useful in practice. While it is true that low precision predictors can be useful in certain contexts, and that lowering precision in order to increase the true positive rate may be desirable depending on your objectives, it is clearly inappropriate in a classification domain to disregard precision completely.

In this paper we demonstrate that when using data with a highly imbalanced class distribution, relying on true positive rates and false positive rates alone (this includes ROC analysis) may provide an overly optimistic view of classifier performance. We demonstrate this by showing that even when pairs of values for these measures appear to be near optimal, there is still considerable room for improvement in practical terms. This is not a novel finding. However, many defect prediction researchers have continued to report their classification results inappropriately since the publication of (Zhang & Zhang 2007). The contribution made here is the intuitive and easily comprehensible presentation of the examples given in Section III.

The rest of this paper is laid out as follows: Section II provides a background to machine learning classifier performance metrics. Section III describes the problem at hand, and why precision is required to appropriately describe classifier performance in highly imbalanced domains. Our conclusions and advice for researchers is presented in Section IV.

II. Background

This section presents an overview of machine learning classifier performance metrics. In this study we limit our scope to that of binary classification problems. For each data point predicted during binary classification, there can be only one of four possible outcomes:

- A *true positive (TP)* occurs when a data point labeled as positive (typically *'defective'* in this domain) is correctly predicted as positive.
- A *true negative (TN)* occurs when a data point labeled as negative (typically *'non-defective'* in this domain) is correctly predicted as negative.
- A *false positive (FP)* occurs when a negative labeled data point is incorrectly predicted as positive.
- A *false negative (FN)* occurs when a positive labeled data point is incorrectly predicted as negative.

These values can be put into a *confusion matrix* (Figure 1).

	labeled positive	labeled negative
predicted positive	TP	FP
predicted negative	FN	TN

Fig. 1. A confusion matrix.

It is worth pointing out that there is a symmetry between the positive and negative classes. However, the positive class typically refers to the class of most interest (*'defective'* modules), which is commonly (in this domain and many others) the minority class.

Useful data statistics and commonly used classifier performance metrics can be derived from a confusion matrix. A subset of these are defined in Table I. Note that in this table, the last two measures defined (*f-measure* and *balance*) are in their most commonly used form. It is however possible to weight them in order to favour either of their comprising measures (Jiang, Cukic & Ma 2008). Additionally note that the *balance* measure was defined in (Menzies, Greenwald & Frank 2007), it is a measurement of distance from a point on a ROC curve to the ideal point, which is typically defined as where the true positive rate is 1 and the false positive rate is 0.

The three measures of most interest in this paper are: the true positives rate, the false positive rate, and precision. The true positive rate describes the proportion of data points labeled as positive which were correctly predicted as such; the optimal value is 1. The false positive rate describes the proportion of data points labeled as negative which were incorrectly predicted as positive; the optimal value is 0. Precision describes the proportion of modules predicted as defective which were correct; the optimal value is 1.

In addition to observing classifier performance with a fixed set of parameters, it may also be desirable to observe how performance varies across a range of parameters. Doing so can be especially beneficial when performing classifier comparisons. ROC curve analysis is commonly used for this task, and involves exploring the relationship between the true positive rate and the false positive rate of a classifier while (typically) varying its *decision threshold*. A trade-off commonly exists between the true positive rate and the false positive rate, and this is demonstrated by ROC analysis via a two dimensional plot of the false positive rate on the x-axis and the true positive rate on the y-axis. The area under the ROC curve (AUC-ROC) is commonly used to summarise a ROC curve in a single measure. The optimal AUC-ROC value is 1.

Precision and recall (PR) curves can be used in the same manner as ROC curves. On a *PR curve*, *recall* is on the x-axis and precision on the y-axis. A trade-off commonly exists between these two measures, and is thus shown on a PR curve. Note that recall is another alias for the true positive rate used in ROC analysis. PR curves are "an alternative to ROC curves for tasks with a large skew in the class distribution" (Davis & Goadrich 2006). The area under the PR curve (AUC-PR) can be computed and used similarly to AUC-ROC.

Alias / Aliases	**Defined As**
Testing Set No. Instances	$TP + TN + FP + FN$
No. Instances in Class 1 (Positive Class)	$TP + FN$
No. Instances in Class 2 (Negative Class)	$TN + FP$
Accuracy **+** Correct Classification Rate 1 - Error Rate	$\frac{TP + TN}{TP + TN + FP + FN}$
Error Rate **-** Incorrect Classification Rate 1 - Accuracy	$\frac{FP + FN}{TP + TN + FP + FN}$
True Positive Rate **+** Recall Sensitivity Probability of Detection (pd) 1 - False Negative Rate	$\frac{TP}{TP + FN}$
True Negative Rate **+** Specificity 1 - False Positive Rate	$\frac{TN}{TN + FP}$
False Positive Rate **-** Type 1 Error Rate Probability of False Alarm (pf) 1 - True Negative Rate	$\frac{FP}{FP + TN}$
False Negative Rate **-** Type 2 Error Rate 1 - True Positive Rate	$\frac{FN}{FN + TP}$
Precision **+**	$\frac{TP}{TP + FP}$
F-Measure **+** F-Score	$\frac{2 * Recall * Precision}{Recall + Precision}$
Balance **+** Distance from ROC optimal point	$1 - \frac{\sqrt{(0-pf)^2+(1-pd)^2}}{\sqrt{2}}$

TABLE I
A SUBSET OF STATISTICS DERIVED FROM A CONFUSION MATRIX. MEASURES MARKED WITH '**+**' HAVE AN OPTIMAL VALUE OF 1. MEASURES MARKED WITH '**-**' HAVE AN OPTIMAL VALUE OF 0.

III. WHY CLASS DISTRIBUTION AFFECTS THE SUITABILITY OF MEASURES

Consider a perfectly balanced data set with 1000 data points, 500 in each class. If we achieve a classification performance of *true positive rate (TPR)* = 1 and *false positive rate (FPR)* = 0.01, it appears as though our classifier has performed very well. All of the data points in the positive class have been correctly classified (TPR = 1), and only 1 percent of data points in the negative class (5 data points) have been incorrectly classified (FPR = 0.01). If we calculate the precision of such a classifier, it works out to 0.99 (to two significant figures). Thus 99 percent of data points predicted to be in the positive class turned out to be correct. In this first example, where we have a balanced class distribution, using the TPR and FPR provided an honest and accurate representation of classifier performance, as a near optimal pair of values likewise resulted in a near optimal precision. A confusion matrix for this example is presented in Figure 2.

	labeled positive	labeled negative
predicted positive	TP = 500	FP = 5
predicted negative	FN = 0	TN = 495

Fig. 2. TPR = 1, FPR = 0.01, Precision = 0.99

Now consider a data set with a highly imbalanced class distribution, 1000 data points in total but with only 10 data points in the positive class (1 percent). If we again achieve TPR = 1 and FPR = 0.01, classifier performance appears to be equal to that of the previous classifier. This is inappropriate, misleading, and exaggerates the classifiers real performance, as the precision of this classifier is 0.50 as opposed to 0.99. Thus, *because of the imbalanced class distribution (i.e. much more data in one class than the other),* the same supposedly near optimal TPR and FPR (or point on a ROC curve) represented vastly different performance. In the first example 99 of every 100 positive predictions were correct, whereas in the second example, the same TPR and FPR represented a classifier where only half of all positive predictions were correct. A confusion matrix for this example is presented in Figure 3.

	labeled positive	labeled negative
predicted positive	TP = 10	FP = 10
predicted negative	FN = 0	TN = 980

Fig. 3. TPR = 1, FPR = 0.01, Precision = 0.50

Turning attention toward the domain of software defect prediction, researchers often use data sets where the minority class represents less than 1 percent of data points in total (see (Menzies, Greenwald & Frank 2007), (Lessmann, Baesens, Mues & Pietsch 2008) and (Jiang & Cukic 2009), for example). We now present an example using the most imbalanced of the NASA Metrics Data Program data sets[1], *PC2*. This data set contains 5589 data points, each consisting of module product metrics and the associated fault data. Just 23 of the data points are labeled as *'defective',* 0.4 percent of data points in total. Thus, with a TPR of 1 all 23 data points labeled as *'defective'* would be correctly classified. An FPR of 0.01 means that 1 percent of the 5566 data points labeled as *'non-defective'* were incorrectly classified. With approximately 56 false positives, a total of 79 modules would be predicted to require further attention. This works out to a precision of 0.29 (to two significant figures), despite the other metrics implying near optimal performance. A confusion matrix for this example is presented in Figure 4.

	labeled positive	labeled negative
predicted positive	TP = 23	FP = 56
predicted negative	FN = 0	TN = 5510

Fig. 4. TPR = 1, FPR = 0.01, Precision = 0.29

[1]http://mdp.ivv.nasa.gov/

In this domain, where the cost of false positives is typically not prohibitively large (see (Menzies, Dekhtyar, Distefano & Greenwald 2007)), such a classifier would, in most environments, be very attractive. Observe however that because of the highly imbalanced class distribution, small changes in the FPR have a large effect on the actual number of false positives. Thus if classifier performance changes to TPR = 1 and FPR = 0.05, the number of false positives increases to 280. The precision in turn drops to 0.08 (to two significant figures). In some environments examining 100 modules to find 8 of them to be defective would not be feasible. A confusion matrix for this example is presented in Figure 5.

	labeled positive	labeled negative
predicted positive	TP = 23	FP = 280
predicted negative	FN = 0	TN = 5286

Fig. 5. TPR = 1, FPR = 0.05, Precision = 0.08

As defect predictors do not achieve performance even close to TPR = 1 and FPR = 0.05, are these theoretical experiments valid? Zhang and Zhang (Zhang & Zhang 2007) indirectly answer this question by highlighting the precision achieved in (Menzies, Greenwald & Frank 2007) on data set *PC2*. Here, despite results appearing to be acceptable (TPR = 0.72, FPR = 0.14), and claims of the naive Bayes classifiers being "demonstrably useful", the precision was just 2.02 percent. The classifier predicted that approximately 796 modules were in the 'defective' class, of which only approximately 17 actually were. This highlights the poor predictive performance achieved, and raises the question of whether such a classifier could be of any practical worth. An optimistic approximation of the confusion matrix for the entire data set (the results in the study were generated after 10 repeated runs of 10-fold cross-validation) is presented in Figure 6.

	labeled positive	labeled negative
predicted positive	TP = 17	FP = 779
predicted negative	FN = 6	TN = 4787

Fig. 6. TPR = 0.74, FPR = 0.14, Precision = 0.02

Note that an identical confusion matrix to the one in Figure 6 can be obtained by simply ranking all data points in data set *PC2* by their *lines of code total* attribute (descending order), and predicting the first $n = 796$ data points only as defective. It is a similar case for 2 more of the 8 data sets used in (Menzies, Greenwald & Frank 2007), where n is the approximate average number of *'defective'* predictions made. This is known as *LOC module-order modelling* (see (Khoshgoftaar & Allen 2003) and (Mende & Koschke 2009)), and highlights both the poor predictive performance of the classifiers, and that research into making defect predictors 'effort aware' is worthwhile (see (Arisholm, Briand & Fuglerud 2007) and (Mende & Koschke 2010)).

Table II presents statistics for each of the 13 NASA Metrics Data Program data sets. These data sets were chosen as they have been heavily used in software defect prediction research.

NASA Data Set Alias	No. Data Points	No. Positive (Minority) Class Data Points	Percentage of Positive Class Data Points	Percentage Precision @ TPR=1, FPR=0.01	Percentage Precision @ TPR=1, FPR=0.05
PC2	5589	23	0.4	29.11	7.64
MC1	9466	68	0.7	41.98	12.64
PC5	17186	516	3.0	75.55	38.22
PC1	1107	76	6.9	88.37	59.38
MW1	403	31	7.7	88.57	62.00
KC3	458	43	9.4	91.49	67.19
CM1	505	48	9.5	90.57	67.61
PC3	1563	160	10.2	91.95	69.57
PC4	1458	178	12.2	93.19	73.55
KC1	2107	325	15.4	94.75	78.50
JM1	10878	2102	19.3	95.98	82.72
MC2	161	52	32.3	98.11	91.23
KC4	125	61	48.8	98.39	95.31

TABLE II
EACH OF THE NASA METRICS DATA PROGRAM DATA SETS RANKED IN ASCENDING ORDER OF PERCENTAGE DATA POINTS IN MINORITY CLASS.

The table shows class distribution details for each data set as well as the precision when TPR = 1, FPR = 0.01 and when TPR = 1, FPR = 0.05. Note that the false positives in these calculations were rounded to the nearest integer. The data sets are ranked in ascending order of the percentage of modules in the positive (minority) class. From the table it can be seen that for the three data sets with the highest class imbalance especially, near optimal values of TPR and FPR results in classifiers which are far from optimal in practical terms, or in terms of precision.

Thus, relying on TPR and FPR 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" (Davis & Goadrich 2006). *Precision is required to give a more accurate representation of true performance in this context.*

IV. CONCLUSIONS

Precision matters, especially when class distributions are highly skewed. When performing any kind of data mining experiment we believe it is very important to document the characteristics of the data being used: where it came from, what pre-processing has been carried out, how many data points and features are present, what is the class distribution, etc. This makes it more accessible for other researchers to check the validity of the claimed results. For example; if such data characteristics are given, it is often possible to derive measures that are not explicitly reported, as done here. The inspiration for the work carried out here came from (Zhang & Zhang 2007), where precision values omitted in (Menzies, Greenwald & Frank 2007) were derived using the TPR, FPR, and class distribution data.

If classifier performance is to be reported with a single set of (hopefully validated and thus suitable) parameters, we believe defect prediction researchers should be reporting a minimum of recall (TPR) and precision, in addition to the data characteristics just described. It is of no harm to also report the false positive rate. Note that it is necessary to take both recall and precision into account when accessing performance, a single one of these measures will not suffice. This is because an optimal recall can be achieved by simply predicting all data points as belonging to the positive class, and an optimal precision can be achieved by only making a single positive prediction, which turns out to be correct. The f-measure (see Table I) is commonly used to combine both measures into a single value, and can simplify performance quantification. When classifier performance is to be reported over a range of parameters, we believe precision and recall (PR) curves to be more suitable in this domain than ROC curves. This is because class distributions are often highly skewed (Menzies, Dekhtyar, Distefano & Greenwald 2007).

Lessmann et al. carried out a large scale benchmarking defect prediction experiment with 22 classifiers (Lessmann et al. 2008). The top 17 of these classifiers were reported to have statistically indistinguishable performance using the AUC-ROC performance measure, and a statistical approach proposed by (Demšar 2006). The data used in the study came from 10 of the NASA Metrics Data Program data sets. Davis and Goadrich in (Davis & Goadrich 2006) point out that with highly imbalanced data sets, PR-curves are more powerful at distinguishing the performance of classification methods than ROC curves. Thus, we think it would be interesting for the experiment by Lessmann et al. to be replicated with PR-curves and AUC-PR.

In a recent paper by Menzies et al. (Menzies, Milton, Turhan, Cukic, Jiang & Bener 2010), it is stated that the "standard learning goal" of defect predictors it to maximize AUC-ROC. We would argue that this should not be the standard learning goal of defect predictors. As shown here, by (Davis & Goadrich 2006) and by (Zhang & Zhang 2007), precision is required in a typically imbalanced domain. Moreover, (Davis & Goadrich 2006) prove "that an algorithm that optimizes the area under the ROC curve is not guaranteed to optimize the area under the PR curve".

In addition to the comments made about defect predictor learning goals, the Menzies et al. paper also states "that accuracy and precision are highly unstable performance indicators for data sets . . . where the target concept occurs with relative infrequency". While it is commonly reported within the data mining literature that accuracy is not suitable for imbalanced data sets, the same can not be said (at all) for precision. This is true other than in the context of experiments to explore the *class imbalance problem* (see (Batista, Prati & Monard 2004)), where the FPR is better suited than precision.

REFERENCES

Arisholm, E., Briand, L. C. & Fuglerud, M. (2007), Data mining techniques for building fault-proneness models in telecom java software, *in* 'ISSRE '07: Proceedings of the The 18th IEEE International Symposium on Software Reliability', IEEE Computer Society, Washington, DC, USA, pp. 215–224.

Batista, G. E. A. P. A., Prati, R. C. & Monard, M. C. (2004), 'A study of the behavior of several methods for balancing machine learning training data', *SIGKDD Explor. Newsl.* **6**, 20–29.

Davis, J. & Goadrich, M. (2006), The relationship between precision-recall and roc curves, *in* 'Proceedings of the 23rd international conference on Machine learning', ICML '06, ACM, New York, NY, USA, pp. 233–240.

Demšar, J. (2006), 'Statistical comparisons of classifiers over multiple data sets', *J. Mach. Learn. Res.* **7**, 1–30.

Jiang, Y. & Cukic, B. (2009), Misclassification cost-sensitive fault prediction models, *in* 'Proceedings of the 5th International Conference on Predictor Models in Software Engineering', PROMISE '09, ACM, New York, NY, USA, pp. 20:1–20:10.

Jiang, Y., Cukic, B. & Ma, Y. (2008), 'Techniques for evaluating fault prediction models', *Empirical Softw. Engg.* **13**(5), 561–595.

Khoshgoftaar, T. M. & Allen, E. B. (2003), 'Ordering fault-prone software modules', *Software Quality Control* **11**, 19–37.

Lessmann, S., Baesens, B., Mues, C. & Pietsch, S. (2008), 'Benchmarking classification models for software defect prediction: A proposed framework and novel findings', *Software Engineering, IEEE Transactions on* **34**(4), 485–496.

Mende, T. & Koschke, R. (2009), Revisiting the evaluation of defect prediction models, *in* 'Proceedings of the 5th International Conference on Predictor Models in Software Engineering', PROMISE '09, ACM, New York, NY, USA, pp. 7:1–7:10.

Mende, T. & Koschke, R. (2010), 'Effort-aware defect prediction models', *Software Maintenance and Reengineering, European Conference on* **0**, 107–116.

Menzies, T., Dekhtyar, A., Distefano, J. & Greenwald, J. (2007), 'Problems with precision: A response to "comments on 'data mining static code attributes to learn defect predictors'"', *IEEE Transactions on Software Engineering* **33**, 637–640.

Menzies, T., Greenwald, J. & Frank, A. (2007), 'Data mining static code attributes to learn defect predictors', *Software Engineering, IEEE Transactions on* **33**(1), 2–13.

Menzies, T., Milton, Z., Turhan, B., Cukic, B., Jiang, Y. & Bener, A. (2010), 'Defect prediction from static code features: current results, limitations, new approaches', *Automated Software Engg.* **17**(4), 375–407.

Zhang, H. & Zhang, X. (2007), 'Comments on "data mining static code attributes to learn defect predictors"', *IEEE Trans. Softw. Eng.* **33**, 635–637.