Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Evaluating prediction systems in software project estimation

Martin Shepperd [a,*], Steve MacDonell [b]

[a] Dept. of IS & Computing, Brunel University, Uxbridge, UB83PH, UK
[b] Dept. of Computing and Mathematical Sciences, Auckland University of Technology, Private Bag 92006, Auckland 1142, New Zealand

A R T I C L E I N F O

Article history:
Available online 25 January 2012

Keywords:
Software engineering
Prediction system
Empirical validation
Randomisation techniques

A B S T R A C T

Context: Software engineering has a problem in that when we empirically evaluate competing prediction systems we obtain conflicting results.
Objective: To reduce the inconsistency amongst validation study results and provide a more formal foundation to interpret results with a particular focus on continuous prediction systems.
Method: A new framework is proposed for evaluating competing prediction systems based upon (1) an unbiased statistic, Standardised Accuracy, (2) testing the result likelihood relative to the baseline technique of random 'predictions', that is guessing, and (3) calculation of effect sizes.
Results: Previously published empirical evaluations of prediction systems are re-examined and the original conclusions shown to be unsafe. Additionally, even the strongest results are shown to have no more than a medium effect size relative to random guessing.
Conclusions: Biased accuracy statistics such as MMRE are deprecated. By contrast this new empirical validation framework leads to meaningful results. Such steps will assist in performing future meta-analyses and in providing more robust and usable recommendations to practitioners.

1. Introduction

Being able to predict is a hallmark of any meaningful engineering discipline and software engineering is no exception. Researchers have been exploring prediction systems[1] for areas such as cost, schedule and defect-proneness for more than 40 years. And whilst considerable sophistication and ingenuity has been brought to bear on the construction of such systems, empirical evaluation has not led to consistent or easy to interpret results. This matters because it is hard to know what advice to offer practitioners who are — or who ought to be — the major beneficiaries of software engineering research.

There has been an enormous growth in interest and empirical research into building prediction systems in software engineering. Many different techniques have been proposed e.g. statistical methods including regression analysis, instance-based learners including case-based reasoners, Bayesian classifiers, support vector machines and ensembles of learners. For an overview see the 2007 mapping study by Jørgensen and Shepperd [16] which identified more than 300 journal papers that examined cost or effort prediction (and this number has continued to grow and, of course, excludes conference publications). Other topics such as defect prediction have generated as much, if not more, attention. It is self-evident that there is a large body of research work.

Given that there are many competing prediction techniques many researchers have set about empirically comparing their performance on different data sets. Unfortunately, not only does no single prediction technique dominate, but there are many contradictory results [34]. To help make more sense of these varied results there has been a recent move to pooling results through systematic reviews and meta-analyses. However, we still tend to find inconclusive results from systematic reviews (or meta-analyses) [19]. Three such examples of inconsistent systematic review findings are:

- Jørgensen [13] reviewed 15 studies comparing model-based to expert-based estimation. Five of those studies found in favour of expert-based methods, five found no difference, and five found in favour of model-based estimation.
- Mair and Shepperd [25] compared regression to analogy methods for effort estimation and similarly found conflicting evidence. From a total of 20 empirical studies, seven favoured regression, four were indifferent and nine favoured analogy.
- Kitchenham et al. [21] found seven relevant empirical studies for the question is it better to predict using local, as opposed

* Corresponding author.
E-mail address: martin.shepperd@brunel.ac.uk (M. Shepperd).

[1] By a prediction system we mean some $f(\mathbf{x}_i)$ to estimate the variable y_i where $\mathbf{x}_i$ is an input vector that describes characteristics of the target i. It need not be formal in the sense of being defined by explicit rules so estimation by humans might be included in this definition. Nor need such systems be deterministic, however, it is required that a prediction system utilises information contained within $\mathbf{x}_i$ and this distinguishes it from guessing at random. In other words prediction systems must, by definition, perform better than random.

0950-5849/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2011.12.008

to cross-company, data. Three studies reported it made no significant difference, whilst four found it was better.

In order to make progress in our research software engineers need to explore the underlying reasons for these inconsistencies and how this unwelcome situation might be resolved. This is extremely important as otherwise it is difficult to make safe recommendations to practitioners. However, I do wish to stress the purpose of this paper is to consider *how best to compare competing prediction* systems, not to argue in favour of any particular prediction technique.

The remainder of this paper is organised as follows. The next section describes a formal framework to provide a context within which to analyse empirical results. I show how randomisation techniques can provide a baseline for interpreting individual primary studies. This serves two purposes. First it can determine the likelihood of a reported level of accuracy not being due to chance. Second, it can be used as an input to calculate the effect size of any change in accuracy relative to chance. Section 3 uses three published, refereed studies [35,36,18] as examples to show how the framework enables unsafe conclusions to be uncovered. These three studies are not intended as a random sample, but rather they are chosen to illustrate that validation problems exist in empirical software engineering and how they may be remedied. In the Discussion Section we conclude that this framework for empirical evaluation of prediction systems provides a basis for rigorous appraisal of results and their significance plus a means of visually combining and interpreting multiple results.

2. A validation framework

In this paper the discussion is restricted to predicting some continuous[2] output that is denoted Y. However, in principle the arguments also apply to classifiers, that is prediction systems where the output is categorical e.g. the module does or does not contain defects. The reason for this distinction is that for accuracy assessment continuous prediction systems deal with residuals [30] whilst classifiers deal with confusion matrices [8].

In order to bring some generality to our discussion and to avoid becoming bogged down with the minutiae of individual studies we propose the following framework. Researchers validate some prediction system P_i over a data set D using some accuracy statistic S according to a validation scheme V. Empirical evaluation can be seen as an attempt to establish an order (or partial order) from binary preference relations such as $P_1 \prec P_2$ over the set P of candidate prediction systems. The preference relation may be read as P_2 is preferred to P_1 or P_1 is less preferable than P_2. It is also useful to combine an indifference relation $\sim$ with a preference relation so one might re-express the previous relation as a non-strict order, thus $P_1 \preccurlyeq P_2$ denotes that P_2 is not worse than P_1 (for a more detailed overview see Davey and Priestley [6]).

The validation scheme V, irrespective of the specific choice of accuracy statistic, can be thought of as an estimator[3] of S. In other words, $\widehat{S}$ is the best guess of the population or true (but generally unknowable) value of S. It is an estimate because, usually it is not practical to try out a prediction system on all software projects, moreover in practice we are most concerned with predicting *future* projects. Therefore researchers need to simulate how the prediction system would behave when dealing with new unseen cases by "holding out" some cases within D to test its ability to predict.

[2] Strictly speaking we also include the absolute scalar type i.e. counting.

[3] An estimator is a statistical procedure for estimating some population parameter from a sample.

The estimator uses rules such as a leave-one out scheme or an $m \times n$ cross-validation. For a discussion and empirical analysis of cross-validation see Kohavi [22]. Although this might seem rather arcane, a study by Song et al. [37] illustrates how important using an unbiased estimator is. They reveal that a previous study reported defect prediction system accuracy results that were the reverse of those obtained when a better validation scheme (one that preserved the integrity of the hold-out sample) was deployed.

More problematic is how we interpret the meaning of the data set D used for validation. Although this is not the usual stance of researchers, it must be seen as a sample drawn from some underlying population over which we wish to say something about S. Clearly our data sets are *not* random samples since this would imply that all projects have an equal chance of being drawn. Another difficulty is the tendency of researchers to avoid making any explicit statement about the population under consideration. Does the researcher mean all software projects? All large projects? All non-student projects? This is an area that needs urgent attention.

When establishing these preference relations researchers need to be concerned with three fundamental questions. For a given accuracy statistic S and candidate prediction systems P_1 and P_2 one must ask:

1. Does the prediction system P_i outperform a baseline of random guessing, a special case of a prediction system denoted P_0, that is does $P_0 \prec P_i$? If the answer is not yes then it cannot even be claimed that P_i is predicting at all since it does worse than random.
2. Is the difference $P_1 \prec P_2$ statistically significant for some predetermined value of α? In other words how likely is any observed effect to have occurred by chance?
3. Is the effect size large enough to justify $P_1 \prec P_2$ in practice? It may be that any improvement that P_2 offers is so inconsequential as to not be worth the effort hence $P_1 \preccurlyeq P_2$ or in other words despite the potential additional effort and sophistication all that can be asserted is P_2 is not worse than an existing P_1.

2.1. Baselines

Generally the notion of some fundamental baseline or benchmark has been absent from validation studies of prediction systems, which is not to say researchers have not made comparisons between competing approaches. However, the interpretation depends upon the choice of approaches which is generally *ad hoc*.

Examples of studies that have employed a baseline are Jørgensen [12] who used sample mean productivity multiplied by estimated size as a fairly simple benchmark to compare the performance of ten other software maintenance effort prediction systems. Interestingly this baseline approach did not always perform worst. However, it still makes some assumptions about measuring size and productivity so it is more a competing prediction system than a fundamental baseline. Another example is, Mendes and Kitchenham [29] who use the sample median as a benchmark for their analysis. Likewise Bi and Bennett [1] suggest the use of the sample mean as the baseline for their proposed anologue of the ROC curve, namely a regression error characteristic curve.

A more naïve and general approach is simply to randomly assign the y value of another case to the target case. This is a form of permutation and has the advantage of not requiring any parameter estimates. We refer to this as random guessing. Any prediction system *should* outperform random guessing over time; to do otherwise calls into question the systematic nature of the prediction system. An inability to predict better than random implies the