Information and Software Technology 73 (2016) 16–18

Contents lists available at ScienceDirect

Information and Software Technology

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

Exact Mean Absolute Error of Baseline Predictor, MARP0

William B. Langdon*, Javier Dolado, Federica Sarro, Mark Harman

CREST, Department of Computer Science, University College, London, Gower Street, London WC1E 6BT, UK

A R T I C L E I N F O

Article history:
Received 17 August 2015
Revised 27 November 2015
Accepted 9 January 2016
Available online 18 January 2016

Keywords:
Software engineering
Prediction systems
Empirical validation
Randomisation techniques
Search based software engineering

A B S T R A C T

Shepperd and MacDonell "Evaluating prediction systems in software project estimation". Information and Software Technology 54 (8), 820–827, 2012, proposed an improved measure of the effectiveness of predictors based on comparing them with random guessing. They suggest estimating the performance of random guessing using a Monte Carlo scheme which unfortunately excludes some correct guesses. This biases their MAR_{P_0} to be slightly too big, which in turn causes their standardised accuracy measure SA to over estimate slightly. In commonly used software engineering datasets it is practical to calculate an unbiased MAR_{P_0} exactly.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Shepperd and MacDonell recently reported problems with often used measures of performance prediction used in software engineering [3]. In particular they report mean magnitude relative error MMRE "will be biased towards prediction systems that underestimate" ([3], page 822) and so they recommend MMRE not be used. Instead they propose standardised accuracy measure SA be used instead:

$$\text{Standardised Accuracy} = \mathrm{SA} = 1 - \frac{\mathrm{MAR}}{\mathrm{MAR}_{P_0}} \times 100$$

Where MAR is the mean of the absolute error for the predictor of interest. E.g. for software project estimation, the average of the absolute difference between the effort predicted and the actual effort the project took. To allow easy comparison they normalise MAR by dividing it by the same measure for random guessing (MAR_{P_0}). They suggest calculating MAR_{P_0} by taking the average error of 1000 runs of random guessing. Instead we show (Eq. 1) the correct exact value can be calculated immediately. Nevertheless Fig. 2 makes clear typically the average of 1000 runs converges to be very close to the long term average. They define random guessing prediction as returning one *of the other* actual measurements of project effort chosen at random. Notice they do not consider that random prediction might stumble across the correct answer by chance. This increases their estimate of the average error. Therefore their Monte Carlo estimate of MAR_{P_0} converges towards a value higher than it should be. The difference is small, their MAR_{P_0} is on average $n/(n-1)$ times higher than it should be. In the case of their Atkinson-2 data set their MAR_{P_0} will be on average 7% higher than it should be. For this particular dataset and prediction technique, correcting the bias in MAR_{P_0} would lead to the predictor's standardised accuracy (SA) being about 7% lower.

As they define MAR_{P_0}, as a result of random sampling, there will always be some variation due to noise. To avoid this complicating subsequent analysis their random MAR_{P_0} should be estimated only once per dataset.

Unfortunately it is common in software engineering prediction experiments to have only a few datasets [3] some of which may be quite small. For most software engineering prediction datasets it is feasible to calculate exactly the average absolute error for such a random guess predictor. This is because for data sets with n outcomes the exact value of MAR_{P_0} can be found in $O(n^2)$ steps. Indeed if there are 2000 or fewer results (e.g. software projects) in the dataset it is faster (needs fewer residuals) to calculate MAR_{P_0} exactly, rather than use a Monte Carlo estimate as suggested by ([3], page 822). For the Atkinson-2 data set, Fig. 2 shows only 120 calculations of abs (measurement vs. random prediction) are needed to get the exact value (263.508), rather than 1000 runs (potentially each uses n abs() steps) to get a value 7% bigger than it should be.

2. Conclusions

The mean absolute error of a predictor which randomly guesses is essential for the normalisation of the standardised accuracy measure SA proposed by Shepperd and MacDonell [3]. However the calculation they proposed is stochastic and biased. The exact

* Corresponding author.
E-mail address: w.langdon@cs.ucl.ac.uk (W.B. Langdon).

http://dx.doi.org/10.1016/j.infsof.2016.01.003
0950-5849/© 2016 Elsevier B.V. All rights reserved.

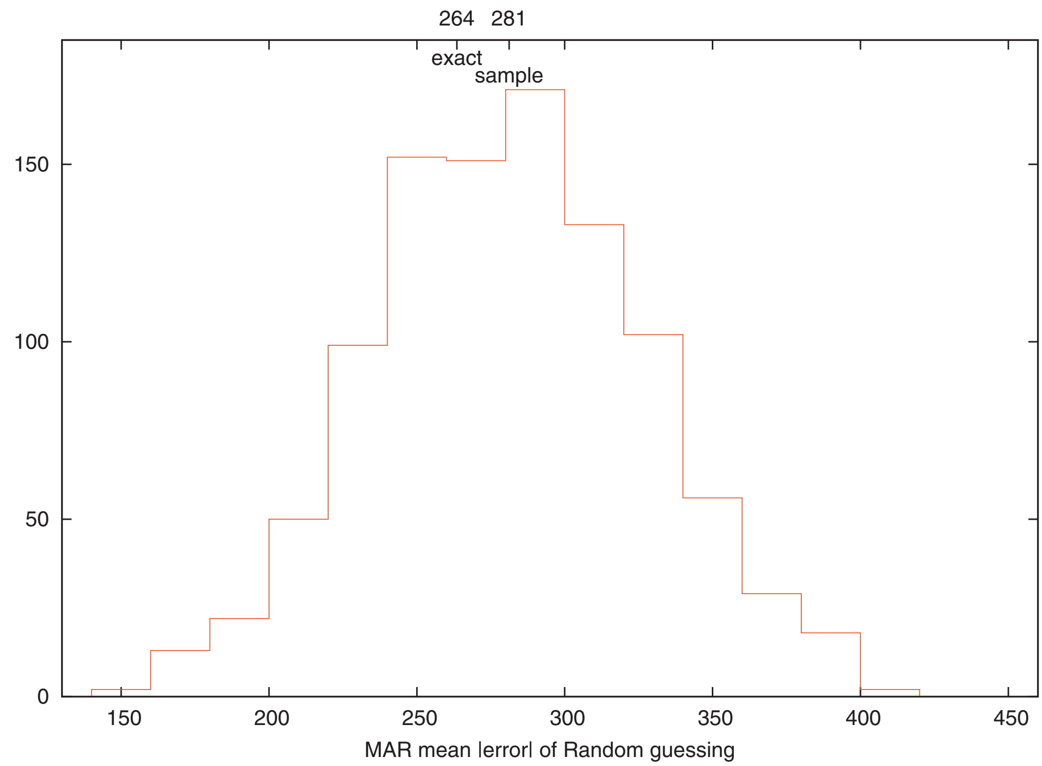

Fig. 1. Histogram of 1000 MAR_{P_0} values from naive guessing of the Atkinson-2 ([2], Table 11) data set. (New random sample.) Notice [3] tends to suggest random guessing (sample mean 281.2) performs worse than it actually does (exact mean 263.508). The variance is 2118 (SD 46) giving a standard error of 1.46 (plotted as the error bar in Fig. 2). .

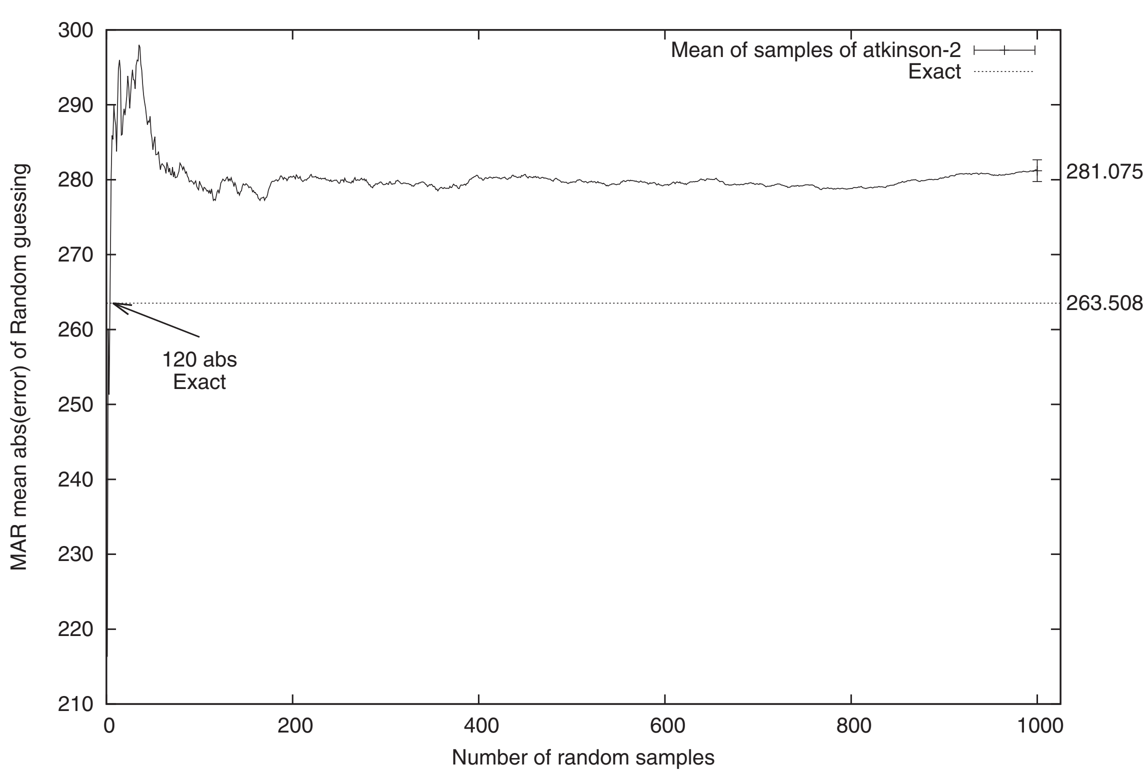

Fig. 2. Only $n(n-1)/2$ (n is size of dataset) calculations are needed to find exact mean absolute error for random guessing. Whereas [3] recommend 1000 samples (each of n calculations), which converges to an overestimate of MAR_{P_0}. (Same data as Fig. 1.) Error bar shows expected variation (i.e. standard error) after 1000 samples.

Table A.1
Mean absolute error of random prediction MAR_{P_0} [3] for the latest version (Nov 2015) of the PROMISE public software engineering effort estimation benchmarks.

Benchmark	Records	Attributes	Predicted attribute	Units	MAR of random prediction
Kitchenham	145	10	Actual.effort		3771.66
Kitchenham	132	10	Actual.effort	(Exclude missing data)	3961.26
Miyazaki94	48	9	MM	Man-Months	111.465
Kemerer	15	8	EffortMM		209.498
China	499	19	Effort		4915.13
Albrecht	24	8	Effort		24.3396
Maxwell	62	27	Effort		8661.64
Cocomo-sdr	12	25	ACTUAL EFFORT	Man month	5.86667
Nasa93	93	24	act_effort	(one month=152 hours)	840.433
Coc81	63	19	actual		1100.47

calculation

$$MAR_{P_0} = \frac{2}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{j<i} |y_i - y_j| \quad (1)$$

avoids the bias by allowing random guesses to alight on the correct answer (i.e., $|\cdot| = 0$ when $i = j$) and is actually faster (i.e. fewer $|y_i - y_j|$ residuals are needed) than a naive Monte Carlo estimate of MAR_{P_0} when $n(n-1)/2 < 1000n$, i.e. when $n <= 2000$. Since the calculation is exact, there are no issues associated with stochastic variations.

Except for large data sets, when calculating SA, the unbiased exact version of MAR_{P_0} should usually be used.

Acknowledgements

I would like to thank Martin Shepperd for helpful suggestions and encouragement.

Appendix A. Precalculated MAR_{P_0} for popular datasets

Although it is straight forward to calculate MAR_{P_0} with Eq. (1), in Table A.1 we provide MAR_{P_0} for some commonly used software engineering prediction benchmarks (see also gawk script in http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/gp-code/exact_marp0.tar).

We downloaded[1] all the public software effort estimation datasets from tera-Promise. (PROMISE is one of the largest repositories specialising in software engineering research datasets [1]).

There are a variety of ways of handling missing data. Two common approaches are (1) supply defaults for or ignore the missing attributes and (2) exclude cases where one or more data are unknown. Accordingly in Table A.1 (the two kitchenham rows) we provide MAR_{P_0} for both approaches (notice difference in number of records).

References

[1] T. Menzies, R. Krishna, D. Pryor, The Promise Repository of Empirical Software Engineering Data, North Carolina State University, Department of Computer Science, 2015.

[2] M. Shepperd, M. Cartwright, A replication of the use of regression towards the mean (R2M) as an adjustment to effort estimation models, Proceedings of the 11th IEEE International Software Metrics Symposium (Metrics 2005), Como, Italy, 2005, pp. 10–38. http://dx.doi.org/doi:10.1109/METRICS.2005.5

[3] M. Shepperd, S. MacDonell, Evaluating prediction systems in software project estimation, Inf. Software Technol. 54 (8) (2012) 820–827. http://www.sciencedirect.com/science/article/pii/S095058491200002X

[1] http://openscience.us/repo/effort/