Multi-stream Online Transfer Learning for Software Effort Estimation: Supplementary Material

Leandro L. Minku
l.l.minku@cs.bham.ac.uk
School of Computer Science, The University of Birmingham
Birmingham, UK

1 SUPPLEMENTARY MATERIAL CONTENT

This file provide larger versions of some figures from the paper [11], and additional information on statistical tests, effect sizes and parameter choices. Specifically, Table 1 presents the effect sizes of the differences in predictive performance of each approach with respect to OATES. Figures 1 and 2 correspond to Figures 1 and 2 from the paper, but with larger size. Figure 3 shows the visualisation of the results of the statistical tests performed to answer RQ3 and relates to Section 6.4 of the paper. The figures are provided from the next page of this report onwards, to enable a higher resolution. Additional information about parameter choices to complement the information provided in Sections 5 and 7 of the paper is given in Section 2. A brief summary of existing work on machine learning for Software Effort Estimation (SEE) is also provided in Section 3 to complement the related work discussed in Section 2 of the paper. OATES' code is available at [10].

2 ADDITIONAL INFORMATION PARAMETER CHOICE

The parameter values listed below were investigated for each base learning algorithm, leading to 20 different combinations:

- Regression tree: minimum number of examples in a leaf node $\in \{1, 2, 3, 4, 5\}$, minimum proportion of the data variance at a node for splitting to be performed $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$.
- k-Nearest Neighbour: $k \in \{1, \cdots, 20\}$.
- Linear Regression, including in the log scale: $ridge \in \{1, 0.5, 10^{-1}, 10^{-2}, 10^{-3}, \cdots, 10^{-18}\}$.

For each dataset, the parameter combination that led to the best median of the n MAE_t measurements for WC $P = 1$ was selected to be used with all approaches.

The following values were investigated for OATES and Dycom's parameters: $lr \in \{0.01, 0.05, 0.1, 0.15\}$ and $\beta \in \{0.3, 0.5, 0.7, 0.9, 1.0\}$. This also leads to $4 \times 5 = 20$ combinations. For SL $P = 1$, 20 different window sizes starting from 10 were investigated. Increments of 1, 2 and 4 were used for the smallest (ISBSG2001 and ISBSGLess), medium (ISBSG2000) and large (ISBSG) datasets. These increments mean that the maximum window size is restricted to $\lfloor n/2 \rfloor$. Larger window sizes would mean that the SL approach is behaving as WC $P = 1$ more than half of the time. For SL $P = \lfloor n/6 \rfloor$, all possible window sizes were used, i.e., $\{1, \cdots, 5\}$. Larger window sizes are not applicable because the number of WC training projects for this approach is 6. For each dataset, the parameters and base learners leading to the best median of the n MAE_t measurements were chosen for the analysis.

$$MAE_t = \frac{1}{min(n', t)} \sum_{i=max(t-n'+1,1)}^{t} |\hat{y}_i - y_i|;$$

$$MLogAE_t = \frac{1}{min(n', t)} \sum_{i=max(t-n'+1,1)}^{t} |log(\hat{y}_i) - log(y_i)|,$$

where $\hat{y}_i$ is the estimation given to the WC project requested to be estimated at time step i, whose true effort is y_i; and t is the current time step.

3 RELATED WORK ON MACHINE LEARNING FOR SOFTWARE EFFORT ESTIMATION

Machine learning for SEE has been studied for many years [2, 4–6, 13, 14, 17]. Existing work has investigated a variety of machine learning algorithms, including linear regression, neural networks, regression trees, k-nearest neighbours, linear programming, etc. As the size of the SEE training sets is typically relatively small, SEE models with too many internal parameters are less likely to perform well [8]. Moreover, the fact that the training sets are heterogeneous makes local learning algorithms such as regression trees, k-nearest neighbours and some cluster-based approaches competitive [1, 7, 9, 12, 14]. Linear regression (potentially applied after log transformations) can also obtain competitive results when the SEE datasets are relatively linear [2, 18]. Ensembles were found to boost the predictive performance of single SEE models [7, 12]. Recently, linear programming has been proposed as a baseline for SEE due to its competitive predictive performance and robustness to different data splits [13]. Some existing work also investigated SEE in the Agile context [3, 15, 16].

Table 1: Effect Size A12 With Respect To OATES' MAE and MLogAE Across Time Steps

Approach	ISBSG2000		ISBSG2001		ISBSG		ISBSGLess	
	A12-MAE	A12-MLogAE	A12-MAE	A12-MLogAE	A12-MAE	A12-MLogAE	A12-MAE	A12-MLogAE
Dycom	0.52	*0.62	0.55	*0.57	0.51	0.53	*0.56	**0.67
SL P=1	*0.56	***0.8	*0.6	**0.69	*0.56	***0.71	-0.52	0.51
WC P=1	0.53	***0.78	***0.73	***0.73	0.54	0.53	*-0.61	0.54
SL P/6	**0.66	***0.81	***0.78	***0.78	**0.64	***0.77	*0.58	***0.75
WC P/6	**0.65	***0.81	***0.91	***0.87	**0.64	***0.77	*0.59	***0.76
Median	**0.66	***0.81	***0.95	***0.93	**0.65	***0.78	*0.6	***0.84

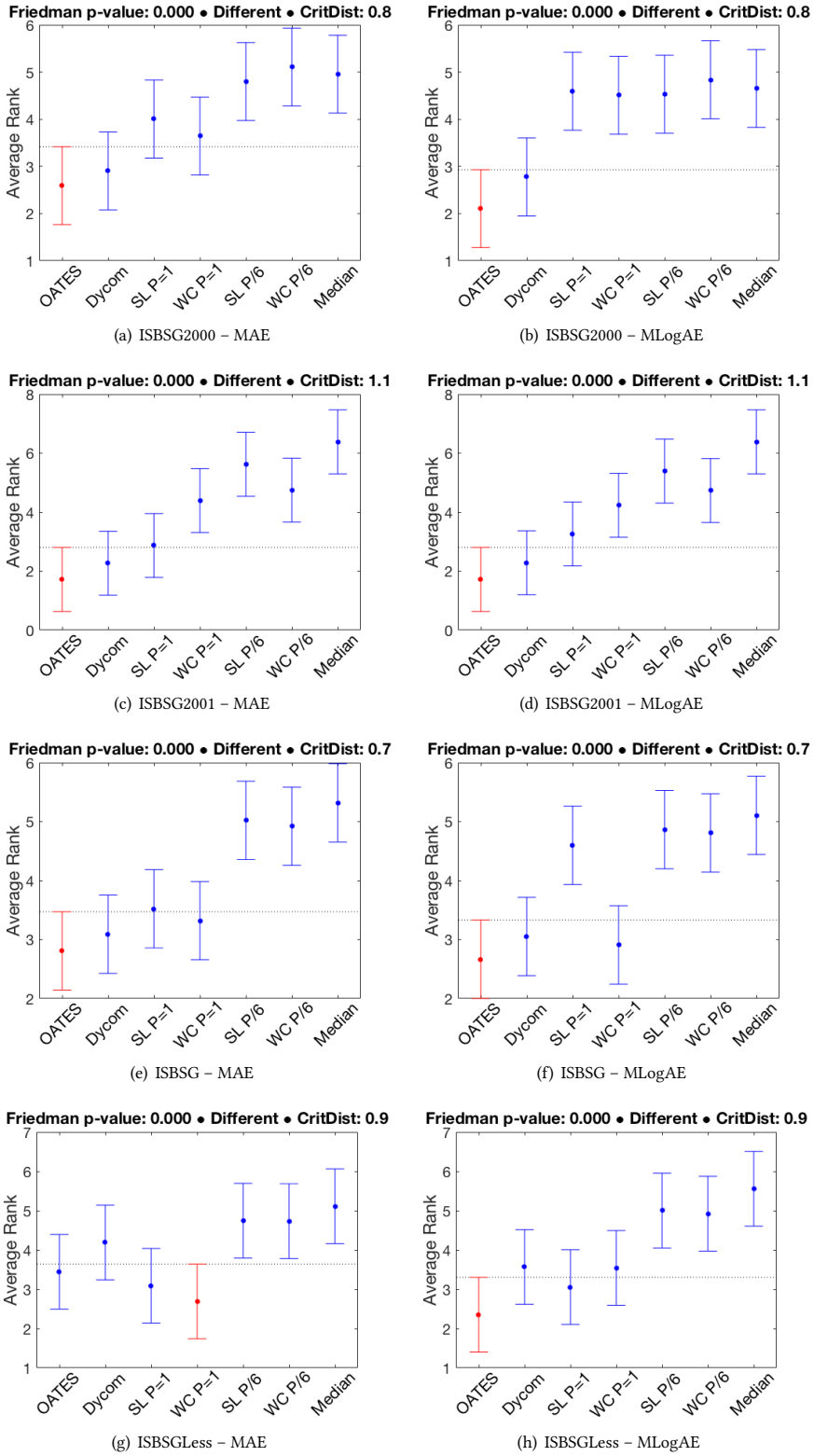
Cells with *, ** and *** indicate small, medium and large effect size, respectively. Cells in orange (dark grey) indicate statistically significant difference w.r.t. OATES according to the Nemenyi tests shown in Fig. 1. Positive A12 indicates values in favour of OATES.

ACKNOWLEDGMENTS

This work was supported by EPSRC Grant No. EP/R006660/2.

REFERENCES

[1] N. Bettenburg, M. Nagappan, and A. E. Hassan. 2012. Think Locally, Act Globally: Improving defect and effort prediction models.. In *MSR*. Zurich, 60–69.
[2] B. Boehm. 1981. *Software Engineering Economics*. Prentice-Hall, NJ.
[3] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. 2019. A Deep Learning Model for Estimating Story Points. *IEEE Transactions on Software Engineering* 45, 7 (2019), 637–656.
[4] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens. 2012. Data Mining Techniques for Software Effort Estimation: A comparative study. *IEEE TSE* 38, 2 (2012), 375–397.
[5] M. Jørgensen and M. Shepperd. 2007. A Systematic Review of Software Development Cost Estimation Studies. *IEEE TSE* 33, 1 (2007), 33–53.
[6] B.A. Kitchenham, E. Mendes, and G.H. Travassos. 2007. Cross versus Within-Company Cost Estimation Studies: A Systematic Review. *IEEE TSE* 33, 5 (2007), 316–329.
[7] E. Kocaguneli, T. Menzies, and J. Keung. 2012. On the Value of Ensemble Effort Estimation. *IEEE TSE* 38, 6 (2012), 1403–1416.
[8] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy. 2013. Active learning and effort estimation: Finding the essential content of software effort estimation data. *IEEE TSE* 39, 8 (2013), 1040–1053.
[9] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann. 2013. Local vs. Global Lessons for Defect Prediction and Effort Estimation. *IEEE TSE* 39, 6 (2013), 822–834.
[10] L. Minku. 2021. minkull/OATES: (Version v1.0). Zenodo. http://doi.org/10.5281/zenodo.5068001
[11] L.L. Minku. 2021. Multi-Stream Online Transfer Learning For Software Effort Estimation – Is It Necessary?. In *Proceedings of the 17th International Conference on Predictive Models in Software Engineering (PROMISE)*.
[12] L.L. Minku and X. Yao. 2013. Ensembles and Locality: Insight on Improving Software Effort Estimation. *IST* 55, 8 (2013), 1512–1528.
[13] F. Sarro and A. Petrozziello. 2018. Linear Programming as a Baseline for Software Effort Estimation. *ACM TOSEM* 27, 3 (2018), 12.1–12.28.
[14] M. Shepperd and C. Schofield. 1997. Estimating Software Project Effort Using Analogies. *IEEE TSE* 23, 12 (1997), 736–743.
[15] Rodrigo G. F. Soares. 2018. Effort Estimation via Text Classification and Autoencoders. In *International Joint Conference on Neural Networks*. 1–8.
[16] M. Usman, E. Mendes, F. Weidt, and R. Britto. 2014. Effort Estimation in Agile Software Development: A Systematic Literature Review. In *International Conference on Predictive Models and Data Analysis in Software Engineering (PROMISE)*. 82–91.
[17] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang. 2012. Systematic Literature Review of Machine Learning Based Software Development Effort Estimation Models. *IST* 54 (2012), 41–59.
[18] P.A. Whigham, C. A. Owen, and S. G. MacDonell. 2015. A Baseline Model for Software Effort Estimation. *ACM TOSEM* 24, 3 (2015), 20.1–20.11.



(a) ISBSG2000 – MAE

(b) ISBSG2000 – MLogAE

(c) ISBSG2001 – MAE

(d) ISBSG2001 – MLogAE

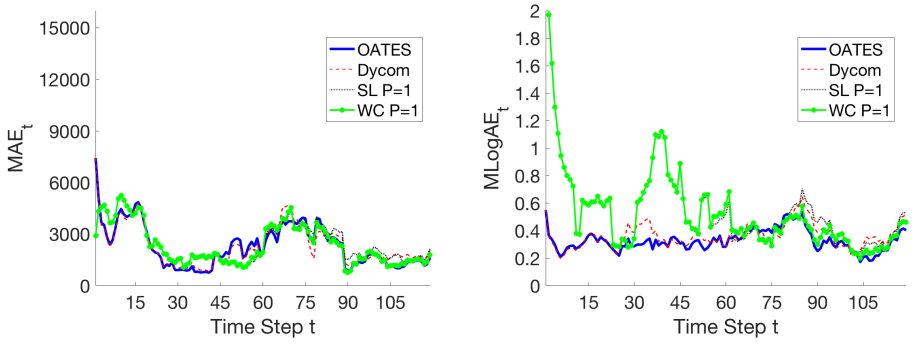
(e) ISBSG – MAE

(f) ISBSG – MLogAE

(g) ISBSGLess – MAE

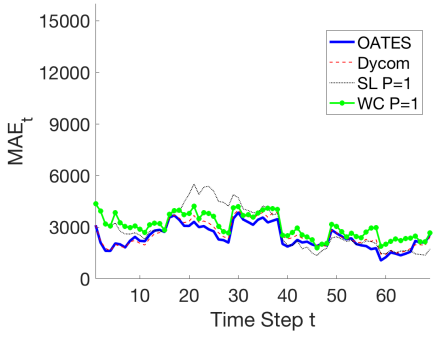
(h) ISBSGLess – MLogAE

Figure 1: Friedman and Nemenyi Tests to Compare Different Approaches in Terms of MAE and MLogAE across time steps. The top ranked approach is shown in red. The dotted horizontal line represents Nemenyi's critical distance with respect to the mean rank of the top ranked approach. Approaches whose mean rank is above this line are significantly different from the top ranked approach.

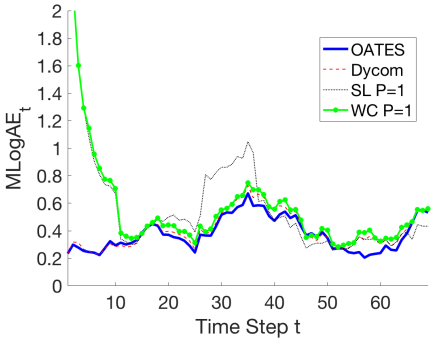


(a) ISBSG2000 – MAE

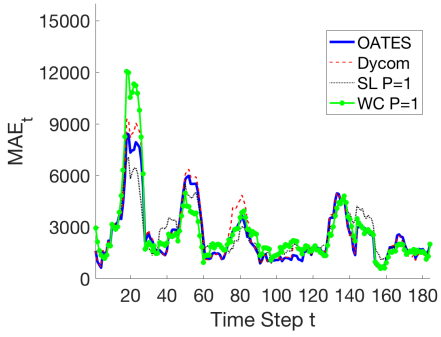
(b) ISBSG2000 – MLogAE



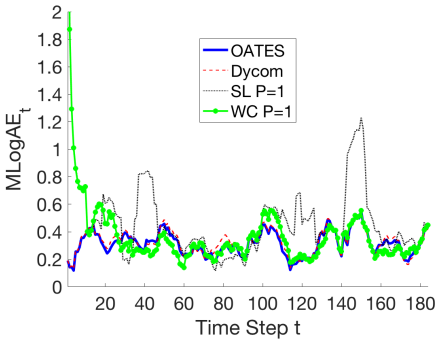
(c) ISBSG2001 – MAE



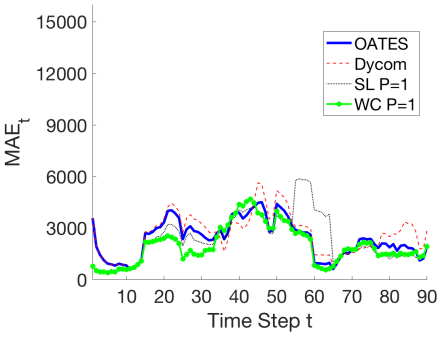
(d) ISBSG2001 – MLogAE



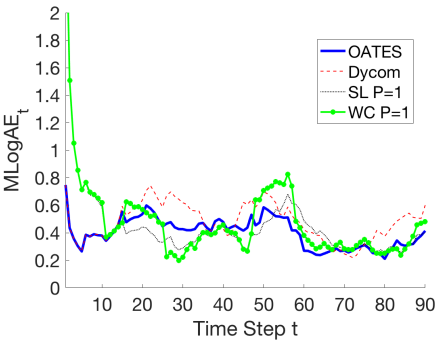
(e) ISBSG – MAE



(f) ISBSG – MLogAE

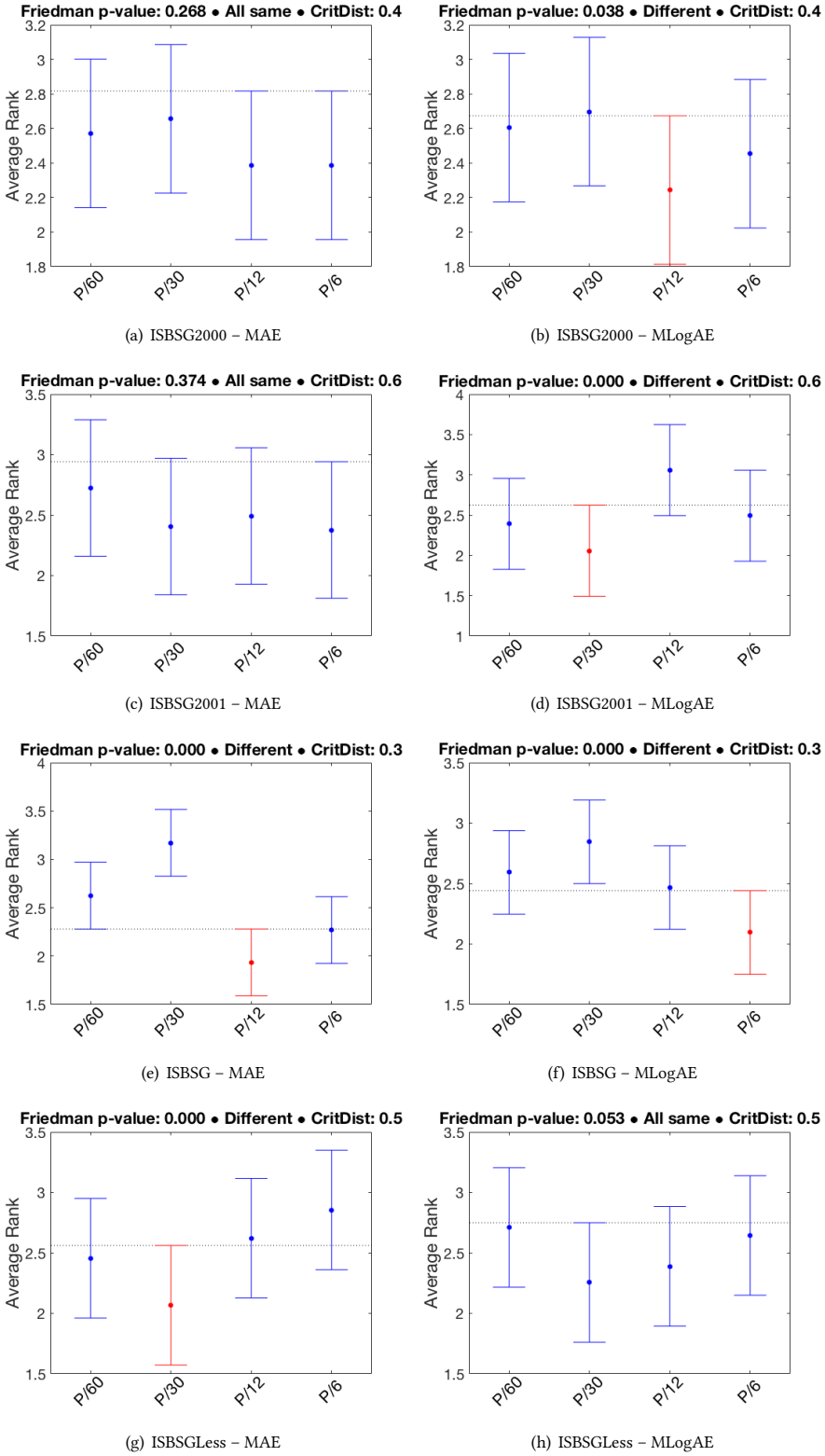


(g) ISBSGLess – MAE



(h) ISBSGLess – MLogAE

Figure 2: MAE and MLogAE Across Time Steps.



(a) ISBSG2000 – MAE

(b) ISBSG2000 – MLogAE

(c) ISBSG2001 – MAE

(d) ISBSG2001 – MLogAE

(e) ISBSG – MAE

(f) ISBSG – MLogAE

(g) ISBSGLess – MAE

(h) ISBSGLess – MLogAE

Figure 3: Friedman and Nemenyi Tests to Compare OATES with Different Values for P in Terms of MAE and MLogAE across time steps. The top ranked approach is shown in red. The dotted horizontal line represents Nemenyi's critical distance with respect to the average rank of the top ranked approach. Approaches whose mean rank is above this line are significantly different from the top ranked approach.