Revisiting the Conclusion Instability Issue in Software Effort Estimation

Michael Franklin Bosu[1], Solomon Mensah[2], Kwabena Bennin[2] and Diab Abuaiadah[1]
[1]Centre for Business, Information Technology and Enterprise, Wintec, Hamilton, New Zealand
[2]Department of Computer Science, City University of Hong Kong, Hong Kong
{michael.bosu, diab.abuaiadah}@wintec.ac.nz, {smensah2-c, kebennin2-c}@my.cityu.edu.hk

***Abstract*—Conclusion instability is the absence of observing the same effect under varying experimental conditions. Deep Neural Network (DNN) and ElasticNet software effort estimation (SEE) models were applied to two SEE datasets with the view of resolving the conclusion instability issue and assessing the suitability of ElasticNet as a viable SEE benchmark model. Results were mixed as both model types attain conclusion stability for the Kitchenham dataset whilst conclusion instability existed in the Desharnais dataset. ElasticNet was outperformed by DNN and as such it is not recommended to be used as a SEE benchmark model.**

Keywords - Conclusion Instability; Software Effort Estimation; Prediction model; ElasticNet; Deep Neural Network

I. INTRODUCTION

Software effort estimation (SEE) is part of the broader discipline of empirical software engineering (EMSE) that rely on evidence to make predictions about the estimated effort required to complete software projects. The importance of predicting software effort cannot be overemphasized as it has effect on the estimated budget for software projects. Both the research community and software engineering practitioners have not been able to identify a frontrunner algorithm as several factors such as datasets, experimental team, pre-processing techniques, etc [1] are known to affect the outcome of these algorithms. The challenge therefore is the ability to consistently and uniformly present the results of empirical software engineering models. The current research evidence indicates several conflicting and confusing results especially with regard to the validity of results as it changes with the aforementioned factors. This phenomenon is known as conclusion instability. Conclusion instability refers to the lack of consistent results observed from software engineering experiments [1]. The cause of this effect is attributed to multiple factors such as preprocessing, different datasets, researcher bias, inadequate reporting of research protocol and so on.

The conclusion instability problem in empirical software engineering unfortunately defeats or it is at variance to the goal of science in general which is the ability of an effect to be observed in multiple experimental conditions [1]. This has affected the generalization of results leading to the discovery of the "best" effort estimation algorithm being elusive.

There are two objectives for this study. First is to assess the viability of an approach introduced in this paper to address "the result interpretation issue" which is an aspect of the conclusion instability problem. This will be done by assessing whether the use of different prediction algorithms on a given dataset will yield the same results. The second objective is to evaluate ElasticNet as a benchmark SEE algorithm. Two classical SEE datasets have been used in the development of effort prediction models in this paper. We will adopt the same procedure employed in our previous paper [2] by generating three effort classes using the density quantile function.

This study applies two SEE modelling techniques on each dataset to assess whether the results are classified into the same classification band. The intention is to assess whether results from multiple estimation models using the same dataset can be classified as generating the same "good" or "bad" results. The advantage of this is that, there will be no need to apply effect size on the models, thus making it easier to interpret and removing one layer of computation because effect size in itself can also lead to conclusion instability when the modelling technique is changed [1]. It is worth noting that we are not comparing estimation models, rather we seek to investigate whether the estimation results of multiple prediction algorithms can be classified into the same band under the same experimental condition (only the dataset changes). We will subsequently assess ElasticNet as a potential benchmarking SEE algorithm with the popular and highly efficient Deep Neural Network (DNN) algorithm.

The rest of the paper is as follows. Section II presents the literature review. Section III is about the method employed in conducting the experiments. Section IV is the analysis of the result and Section V is the discussion and conclusion.

II. LITERATURE REVIEW

Menzies and Shepperd [1] gave prominence to the conclusion instability issue in an editorial of a Special Edition of the EMSE journal in 2012. The authors explained conclusion instability as the inability of software engineering experiments to discover a certain effect which can be reproduced under multiple experimental conditions such as the use of different datasets, algorithms, researchers, accuracy measures and so on. They [1] identified two major sources of conclusion instability as bias and variance. Bias is said to measure the deviation between predicted values and actual values. Variance on the other hand is measured by the deviation between different predictions of the estimators.

Turhan [3] outlined characteristics of SEE data that could result in conclusion instability. Covariate shift is where the distribution of the training set is different from the validation or test sets and as such the model that was generated by the training data is not able to predict the test data effectively as well as any future project. Prior probability shift is where the distribution of the explanatory variable of the training data and test data are

DOI reference number: 10.18293/SEKE2018-126

different. Other types of data shift problems are sample selection bias, imbalanced data, domain shift and source component shift

Menzies et al. [4] conducted a study using 158 SEE methods based on COCOMO dataset features. It was realized that “different datasets sources, different evaluation methods and different random selection of data” led to different results at each occasion which confirm the conclusion instability in SEE methods. They however found four methods that consistently provided better results than the others.

Mair and Shepperd [5] reviewed the result of studies that compared regression techniques with analogy techniques for software cost prediction. They discovered inconsistencies in the result. They found no clear favourite technique as an equal number of studies favoured either approach. They also observed results being inconsistent in cases where even the same datasets were used. This is one of the issues this study is attempting to explain, whether the results were actually inconsistent or there should be a new approach in interpreting the result.

Two research questions (*RQ*) are used to address the objectives of this study:

RQ1: Do SEE models of different learning algorithms result in the same effort class?
RQ2: How does the performance of Deep Neural Network SEE models differ from ElasticNet SEE models?

III. METHODOLOGY

A. Dataset Description

Two classical datasets from the tera-PROMISE repository (http://openscience.us/repo/) have been used in the development of the SEE models. Though these datasets are old, we employed them because they have become the benchmark datasets for SEE studies. A brief description of these datasets is provided.

The **Desharnais** dataset was collected by Jean-Marc Desharnais from ten organizations in Canada. The projects in this dataset were undertaken between 1983 and 1988. The dataset consists of 81 records and 12 attributes, with size measured in function points. We used the 77 version of the dataset as a result of 4 missing records. Summary statistics for relevant features of the dataset are provided in Table 1.

Table 1. Descriptive statistics for Desharnais dataset

Feature	N	Min	Max	Mean	Std.Dev	Skew	Kurt
TeamExp	79	0	4	2.27	1.34	-.042	-1.26
ManagerExp	78	0	7	2.67	1.52	.20	.07
Transactions	81	9	886	179.90	143.32	2.36	7.73
Entities	81	7	387	122.33	84.88	1.34	1.48
Envergure	81	5	52	27.63	10.59	-.11	-.28
PointsNonAjust	81	62	1116	287.05	185.11	1.67	4.16
Effort	81	546	23940	5046.31	4418.77	2.01	4.72

The **Kitchenham** dataset was collected from American-based multinational Computer Sciences Corporation (CSC). This dataset contains information related to 145 software development and maintenance projects that CSC undertook for several clients.

The projects were undertaken between 1994 and 1999 with 10 attributes including the start date and estimated completion dates. Summary statistics of the relevant features of the Kitchenham dataset are provided in Table 2.

Table 2. Descriptive statistics for Kitchenham dataset

Feature	Min	Max	Mean	Std.Dev	Skew	Kurt
Actual_duration	37	946	206.45	134.09	1.93	6.12
AFP	15.36	18137.48	527.67	1521.99	10.92	126.70
Actual_effort	219	113930	3113.12	9598.01	10.87	125.64

B. Data Preprocessing

In order to ameliorate the problem of data quality such as missing data, outliers and inferential data, the two datasets were preprocessed in the following ways for effective and efficient model construction.

We observed all the instances per each dataset to eliminate missing values. Only a handful of missing entries (4 instances) were observed in the Desharnais dataset. Out of the 145 projects in the Kitchenham, three projects were removed as they were found to have missing entries. With the use of kernel density plots and data trimming technique [6], outliers were identified and removed. Cook’s distance was used in the identification and treatment of influential data points during the model construction. In this study, we realized that, these influential data points identified yielded no negative effects on the models when the datasets were normalized using the *z-score* normalization technique as done in our previous study [2].

We selected *prior* features/variables which are known prior to the development of a new software project. Thus, with regard to the Kitchenham dataset, we made use of the *adjusted function points (AFP)* and *project type* as the independent variables and *actual effort* as the dependent variable. With regard to the Desharnais dataset, the independent variables selected are *team experience (teamExp), manager’s experience (managerExp), programming language, number of entities, number of unadjusted function points (pointsNonAdjust), number of transactions* and *development environment* (*envergure).* The dependent variable from the Deshairnais is the development *effort* variable measured in person-hours.

C. Effort Estimation Models

Two prediction models have been applied to the two studied datasets. Deep Neural Network (DNN) and the ElasticNet (ENR) algorithms have been used in the development of the SEE models. ENR was found in our previous study [2] as a viable benchmarking model. The ElasticNet proposed by Zou and Hastie [7] is a regularization and variable selection technique which helps in eliminating highly correlated predictor variables from the estimation model.

We benchmarked the prediction results from the ElasticNet model to a complex and robust prediction model, namely a Deep learning model which yielded better prediction accuracy in a previous study [8]. We constructed a DNN which makes use of multiple hidden layers and an output layer with their respective neurons to automatically learn from a set of project cases and gives the resulting prediction for the target (in our case, the software effort of *new* projects). The Levenberg-Marquardt backpropagation optimization training function is employed to update the weights of the neurons in the hidden and output layers respectively. The hyperbolic tangent activation function is used in each of the neurons for giving the respective outputs. We followed the same experimental setup as done in previous study [2] by using the leave-one-out cross validation for setting up each of the prediction models.

In order to facilitate self-guidance in the interpretation of the level of effort expended, we made use of the classification scheme defined in Mensah et al. [2] to classify the results of our modelling to the appropriate effort class. Software effort classification is the process of categorizing estimated software effort into its respective class. The scheme is based on historical project datasets (historical data being the same as training set in this study). A goal of the classification scheme is to facilitate easy interpretation of the estimated software effort (Y_R) in the context of existing organization data. In order to achieve these levels of classification, we discretize the actual efforts of the datasets into three classes (*low, moderate* and *high*) based on the density quantile theory. This density quantile theory was utilized because it gives rise to an optimal spacing selection threshold values for categorizing the effort into their respective classes [2].

The selected independent variables together with the software effort (dependent variable) are used to setup the prediction models. We denote the estimated effort values from the models as Output (Y_R). The Output (Y_R) from the predictive model is then classified into its respective class. The estimated effort, Output (Y_R) are categorized into their respective effort classifications. Thus, estimated effort values less than Q_l are classified as Low Effort and estimated effort values more than Q_h are classified as High Effort. Lastly, estimated effort values falling within the thresholds [Q_l Q_h] are classified as Moderate.

D. Accuracy Measures

In order to evaluate the effectiveness of the SEE models, Mean Absolute Error (MAE) and Logarithmic Standard Deviation (LSD) accuracy measures have been employed.

MAE is a risk function that measures the average absolute deviation of the estimated effort values from the actual or true effort values. LSD is defined as the root of the average squared sum of the deviations and the variances between the estimated effort and the true effort. LSD uses the residual in the log-scale, which is independent of size (i.e., homoscedastic). The LSD measure of impurity was applied to the datasets. This index is computed as the within-node variance, adjusted for frequency or case weights (if any). MAE and LSD have been recommended by Foss et al. [9] as a robust and reliable performance measures in setting up SEE models.

Aside of the MAE and LSD evaluation measures, we also considered the Yuen's test and the Cliff's delta (δ) effect size measures as statistical and robust evaluation measures [6]. The statistical test was done at 5% significance level. Even though the Yuen's test is a robust test statistic for assessing statistical significance, it is not enough to make accurate assessment of a tested hypothesis [6]. The Cliff's δ effect size is chosen in addition to the Yuen's test since it yields an effective computation measure irrespective of both the experimental and control groups having different sample sizes. Again, it is not affected by outliers and does not assume the sampling data to follow any distribution. The rationale behind the Cliff's δ effect size is that, given two groups of observations without necessarily following the same distribution, this effect size is able to determine the amount of overlap between these two groups.

IV. Results

We discuss the results of the SEE models built in this section. We provide the classification of the software effort values using the density-quantile function for the normalized/un-normalized datasets in Table 3. Three classes (low, moderate and high) have been created.

Table 3. Software effort classification based on Density-quantile function

Dataset	**Normalized data**		
	Low	**Moderate**	**High**
Kitchenham	0 - 0.1295	0.1296 - 0.2628	> 0.2628
Desharnais	0 - 0.2995	0.2996 - 0.8870	> 0.8871
	Un-normalized data		
	Low	**Moderate**	**High**
Kitchenham	0 - 846.2	846.3 - 2.9122	> 2.9122
Desharnais	0 – 2346.8	2346.9 – 6042.8	> 6042.8

RQ1: Do SEE models of different learning algorithms result in the same effort class?

Table 4 presents the recorded losses of the estimated and classified effort from the two learners across each normalized/un-normalized dataset. Here, a '1' denotes a correct classification of the estimated effort and a '0' denotes a wrong classification. It was observed that the estimated effort from the DNN model resulted in correct classification of the effort in both cases of the normalized/un-normalized datasets (Table 4).

Table 4. Classification performance of predicted effort *wrt* to number of losses

Dataset	**Normalized data**		**Un-normalized data**	
	ENR	**DNN**	**ENR**	**DNN**
Kitchenham	1	1	1	1
Desharnais	0	1	0	1

Thus, irrespective of applying data normalization technique, we realized that the DNN resulted in correct classification of the estimated effort values. Table 5 denotes the predicted effort from the two learners for each dataset against the actual effort benchmark.

Table 5. Actual vs. Predicted effort and their respective effort classes

Dataset	**Normalized data**			**Un-normalized data**		
	Actual effort	**Predicted effort**		**Actual effort**	**Predicted effort**	
		ENR	**DNN**		**ENR**	**DNN**
Kitchenham	0.2918 (H)	0.5048 (H)	0.3403 (H)	3113.1 (M)	2214.411 (M)	3128.0 (M)
Desharnais	0.7183 (M)	0.9305 (H)	0.8099 (M)	4833.9 (M)	2203.6 (L)	4494.8 (M)

We found that the DNN resulted in improved prediction accuracy against the ElasticNet regression approach.

For the Kitchenham dataset, both normalized and un-normalized, the two learners, ElasticNet and DNN modeling results were correctly classified into the same respective effort class as shown in Table 5. Only the DNN made accurate classification of the estimated effort into the correct classes for the Desharnais dataset whilst the ElasticNet did not. Thus, the Desharnais dataset can be said to exhibit traits of conclusion instability with respect to the ElasticNet prediction model.

RQ2: How does the performance of Deep Neural Network SEE models differ from ElasticNet SEE models?

We compared the prediction performance of the Elastic and DNN models using MAE and LSD as shown in Table 6. Note that the shaded cells denote the best performance (minimum values) for either the ElasticNet or the DNN in each case of the normalized/un-normalized dataset. For example, with regard to the normalized Desharnais dataset, we found that the DNN yielded improved prediction accuracy on average irrespective of using the MAE or LSD for evaluation. The results from the statistical test in Table 7 show that the *p-values* from the Yuen's test resulted in significant differences between the DNN and the ElasticNet irrespective of the application of data normalization. We found from the Cliff's δ effect size that there exists significant differences between using the DNN and ElasticNet for estimating the software effort in all cases with the exception of the normalized Kitchenham dataset (where δ was 0.255<0.276).

Table 6. Performance Evaluation of Learners using MAE and LSD

Dataset	Learner	Normalized data		Un-normalized data	
		MAE	**LSD**	**MAE**	**LSD**
Kitchenham	ENR	0.2502	0.5531	3096.7	9598.0
	DNN	0.2177	0.9562	2350.9	7310.6
Desharnais	ENR	0.9922	1.1470	4829.7	4187.8
	DNN	0.5148	0.6908	3136.0	4186.2

Table 7. Statistical significant differences of predicted effort across learners

Learner	Normalized data			Un-normalized data		
	Yuen's test		**Cliff's δ**	**Yuen's test**		**Cliff's δ**
	***t*-value**	***p*-value**		***t*-value**	***p*-value**	
Kitchenham Dataset						
DNN *vs.* ENR	6.37	2.2e-08*	0.255	-9.19	9.9e-14*	0.763**
Desharnais Dataset						
DNN *vs.* ENR	4.54	2.7e-05*	0.363**	-23.95	4.4e-32*	0.410**

*Statistical Significance: *p<0.05; Practical Significance: **δ≥0.276*

V. Discussion and Conclusion

In this paper, we built two SEE models, ElasticNet and DNN and applied them to two SEE datasets. The SEE models were executed using both normalized and un-normalized data. The following questions were addressed by the models:

RQ1: Do SEE models of different learning algorithms result in the same effort class?

For the Kitchenham dataset, when normalized and un-normalized data were used, both ElasticNet and Deep Neural Network (DNN) modeling results were classified into the correct effort class as the actual classes of the software effort. Correct effort class classification was however, not achieved by ElasticNet models when applied to the Desharnais dataset. This result is particularly interesting as it demonstrates that by using the classification of efforts values it is possible to address the conclusion instability problem as has been achieved for the Kitchenham dataset. Thus, DNN and ElasticNet can achieve conclusion stability irrespective of whether the data is normalized or not.

RQ2: How does the performance of Deep Neural Network SEE models differ from ElasticNet SEE models?

Although the ElasticNet models proved superior against other linear regression based SEE models [2], it has performed abysmally against the DNN SEE models and as such it cannot be considered for use as a benchmark model for SEE models. Though this is negative result, we report it in alluding to some of the reasons offered by Kocaguneli et al. [10] in reporting negative scientific results. The result reported in this paper should inform researchers of future studies not to benchmark the ElasticNet SEE models against DNN SEE models. It also offers positive knowledge as it establishes the certainty of the result due to the rigorous experimental approach followed in developing the SEE models in this paper.

Future work will apply DNN and other learners to multiple industrial datasets to determine the existence or otherwise of the conclusion instability issue.

References

[1] T. Menzies and M. Shepperd, "Special issue on repeatable results in software engineering prediction," *Empir. Softw. Eng.*, vol. 17, no. 1–2, pp. 1–17, 2012.

[2] S. Mensah, J. Keung, M. F. Bosu, and K. E. Bennin, "Duplex output software effort estimation model with self-guided interpretation," *Inf. Softw. Technol.*, vol. 94, pp. 1–13, 2018.

[3] B. Turhan, "On the dataset shift problem in software engineering prediction models," *Empir. Softw. Eng.*, vol. 17, no. 1, pp. 62–74, 2012.

[4] T. Menzies, O. Jalali, J. Hihn, D. Baker, and K. Lum, "Stable rankings for different effort models," *Autom. Softw. Eng.*, vol. 17, no. 4, pp. 409–437, 2010.

[5] C. Mair and M. Shepperd, "The consistency of empirical comparisons of regression and analogy-based software project cost prediction," *2005 Int. Symp. Empir. Softw. Eng. ISESE 2005*, pp. 509–518, 2005.

[6] B. Kitchenham *et al.*, "Robust Statistical Methods for Empirical Software Engineering," *Empir. Softw. Eng.*, vol. 22, no. 2, pp. 579–630, 2017.

[7] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. R. Stat. Soc. Ser. B (Statistical Methodol.*, vol. 67, no. 2, pp. 301–320, 2005.

[8] S. Mensah, J. Keung, S. G. MacDonell, M. F. Bosu, and K. E. Bennin, "Investigating the significance of bellwether effect to improve software effort estimation," *Proc. - 2017 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2017*, pp. 340–351, 2017.

[9] T. Foss, E. Stensrud, B. Kitchenham, I. C. Society, and I. Myrtveit, "A Simulation Study of the Model Evaluation Criterion MMRE," IEEE Trans. Softw. Eng vol. 29, no. 11, pp. 985–995, 2003.

[10] E. Kocaguneli, T. Menzies, and J. W. Keung, "Kernel methods for software effort estimation. Effects of different kernel functions and bandwidths on estimation accuracy," *Empir. Softw. Eng.*, vol. 18, no. 1, pp. 1–24, 2013.