Software Maintenance Effort Estimation – Neural Network Vs Regression Modeling Approach

Ruchi Shukla
Department of Computer Science and Engineering
Motilal Nehru National Institute of Technology
Allahabad, India – 211004

A K Misra
Department of Computer Science and Engineering
Motilal Nehru National Institute of Technology
Allahabad, India – 211004

ABSTRACT
The global IT industry has now matured. As more and more systems grow old and enter into the maintenance stage, software maintenance (SM) is becoming one of the most carried out and challenging tasks. Besides, the industry is also facing a shift in traditional technical environment by way of use of newer tools and approaches of software development, migration from legacy software to current software and dynamic changes in the SM environment. The challenge then lies in accurately modeling and predicting the SM effort, schedule and risk involved, under the above circumstances. This work employs a neural network (NN) approach to model and predict the software maintenance effort based on an available real life dataset of outsourced maintenance projects (*Rao and Sarda, 36 projects of 14 drivers*). A comparison between results obtained by NN and regression modeling is also presented. It is concluded that NN is able to successfully model the complex, non-linear relationship between a large number of effort drivers and the software maintenance effort, with results closely matching the effort estimated by experts.

Categories and Subject Descriptors

General Terms

Keywords
Software maintenance, Effort estimation, Neural network, Regression.

1. INTRODUCTION
Software is typically delivered with undiscovered flaws. As per the IEEE standard for software maintenance (SM) the definition of SM is as follows: "The modification of a software product after delivery, to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment" [1]. SM today is the most expensive and time consuming phase especially in case of legacy, large and complex systems. Due to architectural modifications their original design no longer matches the new business goals and requirements [2]. SM is a dynamic process and its planning involves estimating size, effort, duration, staff and costs. Problems of maintainer's job switchover, recruitment of experienced maintainers, costing and total project duration while submitting a maintenance bid, optimum resource allocation and vast variety of projects have made accurate estimation of maintenance cost a fairly challenging problem for the maintenance organizations.

2. LITERATURE REVIEW
International Software Benchmarking Standards Group – ISBSG - 2005 *(http://www.isbsg.org*) provides an initial analysis of the ISBSG maintenance and support data. The other popular datasets include - COCOCMO 81 and COCOCMO II, COSMIC, IFPUG, Rao and Sarda, Desharnais, Kemerer etc. ([3], [4]). Recent research has focused on the use of function points (FPs) in effort estimation. However, a precise estimation should not only consider the FPs, representing the software size, but should also include different elements of the development environment. Reference [5] proposed a SM project effort estimation model based on function points. It used FPs to calculate the volume of maintenance function. Ten value adjustment factors were considered and grouped into three categories of maintenance characteristics, i.e. the people domain, product domain and the process domain.

Various mathematical and machine learning or artificial intelligence (AI) based techniques like regression analysis, artificial neural networks (ANN), genetic algorithms (GA), fuzzy logic (FL), case based reasoning etc. are being used for accurate prediction and estimation of SM effort ([6]-[8]). Most of these studies are based on the hard to estimate maintained code size metric 'lines of code' (LOC) or the FP metrics. Reference [9] presented a review of studies on estimation of software development effort. The unit effort expended on maintenance of a system was dependent on many external factors and was not a linear relation with respect to time [10]. Reference [11] compared the prediction accuracy of different models using regression, neural networks and pattern recognition approaches. Reference [12] listed the following four groups of factors affecting the outsourced maintenance effort: system baseline, customer attitude, maintenance team and organizational climate; and described how a system dynamics model could be build.

However, till date no single estimation model has been successfully applied across a wide variety of projects. Although, there are many likely benefits of using more than one technique, there is no way to decide beforehand, which techniques can be

applied for SM effort estimation. Often, adequate information of real life SM projects regarding size, maintenance history, human and management factors (management focus, client attitude, need for multi-location support teams etc.) is unavailable. This makes the problem of objectively estimating SM effort almost in-tractable.

Artificial intelligence combines the elements of learning, adaptation and evolution e.g. NN and FL that are able to learn from experimental data, represent highly non-linear and multi-variate relationships, and are expertise or rule based. These have been successfully applied to an environment typically present in a modern day SM company ([13]-[17]). Many AI based hybrid schemes have also been investigated for SM effort estimation including neuro-GA, grey-GA, neuro-fuzzy, etc. ([18]-[20]). Hence, a soft computing approach based on ANN is preferred in the present work.

3. PROPOSED WORK

The objective of the present work is to develop a multilayer feed forward NN with back-propagation and Bayesian regularization training. The choice of neural networks as the estimation tool was governed by the fact that a properly trained NN gives matching outputs when presented with unseen inputs, as is the case in SM effort estimation. The present work is based on the open literature effort data of 36 outsourced SM projects of 14 effort drivers as shown in Table 1 and Appendix 1 [4]. No NN based SM effort estimation studies using this dataset are available in the literature.

Table 1. Effort Drivers.

Sl.	Effort Drivers
A.	Existence of restart/recovery logic in batch programs
B.	Percentage of the online programs to the total number of programs
C.	Complexity of the file system being used
D.	Average number of lines per program
E.	Number of files (input, output, sort) used by the system
F.	Number of database objects used by the system
G.	Consistency and centralization of exceptional handling in programs
H.	Whether structured programming concepts have been followed in the program
I.	Percentage of commented lines of code to the total lines of code of the system
J.	Number of programs executed as part of a batch job
K.	Number of database structures used by a typical program
L.	% of the update programs to the total number of programs
M.	Nature of service level agreement (SLA)
N.	Whether structured programming concepts have been followed in the program

The organization of rest of the paper is as follows: Section 4 presents the results of statistical analysis and regression modeling. Section 5 deals with the neural network modeling approach adopted in the present work. Section 6 presents the analysis and validation of results obtained while Section 7 presents the concluding remarks.

4. STATISTICAL ANALYSIS AND REGRESSION MODELING

Before conducting regression analysis we proceed to check if the data was normally distributed. Fig. 1 shows a histogram plot of a normally distributed dataset. From the data of effort drivers as input and estimated effort as output, we ranked the 14 effort drivers based on the Taguchi signal-to-noise ratio concept, for the 'smaller-is-better' optimization criterion. A linear regression model (Eq. 1) was obtained using the commercial package Minitab, by conducting S/N ratio based ANOVA (Analysis of Variance), as shown in Table [2]. The obtained P (probability) values gave the relative importance of each variable.

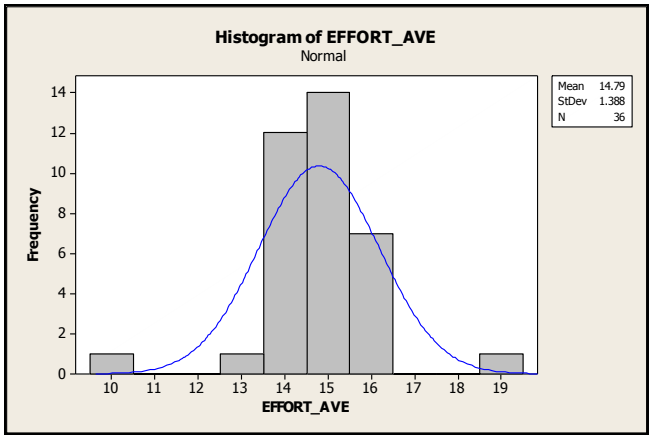

Fig. 1. Histogram showing normal distribution of data.

Table 2. ANOVA Analysis.

Predictor	Coef	SE Coef	T	P
Constant	6.6309	0.5142	12.90	0.000
A	0.3111	0.1414	2.20	0.039
B	0.029167	0.004329	6.74	0.000
C	0.46667	0.08658	5.39	0.000
D	0.00040556	0.0000577	7.03	0.000
E	0.0016500	0.0003463	4.76	0.000
F	0.028611	0.005772	4.96	0.000
G	0.004167	0.004329	0.96	0.347
H	0.03490	0.01920	1.82	0.083
I	0.03264	0.01443	2.26	0.034
J	0.05556	0.01443	3.85	0.001
K	0.04375	0.01443	3.03	0.006
L	0.019928	0.003764	5.29	0.000
M	0.10867	0.01920	5.66	0.000
N	0.39792	0.04329	9.19	0.000

S = 0.424139 R-Sq = 94.4% R-Sq(adj) = 90.7%

EFFORT_AVE = 6.63 + 0.311 A + 0.029 B + 0.467 C + 0.0004 D + 0.0016 E + 0.028 F + 0.0041 G + 0.0349 H + 0.0326 I + 0.0556 J + 0.0437 K + 0.019 L+ 0.109 M + 0.398 N (1)

The parameter N (whether structured programming concepts have been followed in the program) is found to have a considerably dominant effect on the effort and is ranked at no. 1, while the parameter G (consistency and centralization of exceptional handling in programs) has the least significant effect. A high value of 0.944 of the square of correlation coefficient (R-Sq) shows an excellent agreement between the linear model predicted and experimental values, further indicating the consistency of data. Thereafter the main effect plot (Fig. 2) was drawn to evaluate the change in mean effort at different level settings of each variable. It is evident that almost all the drivers except G and H had an increasing effect on the predicted effort with increased level settings.

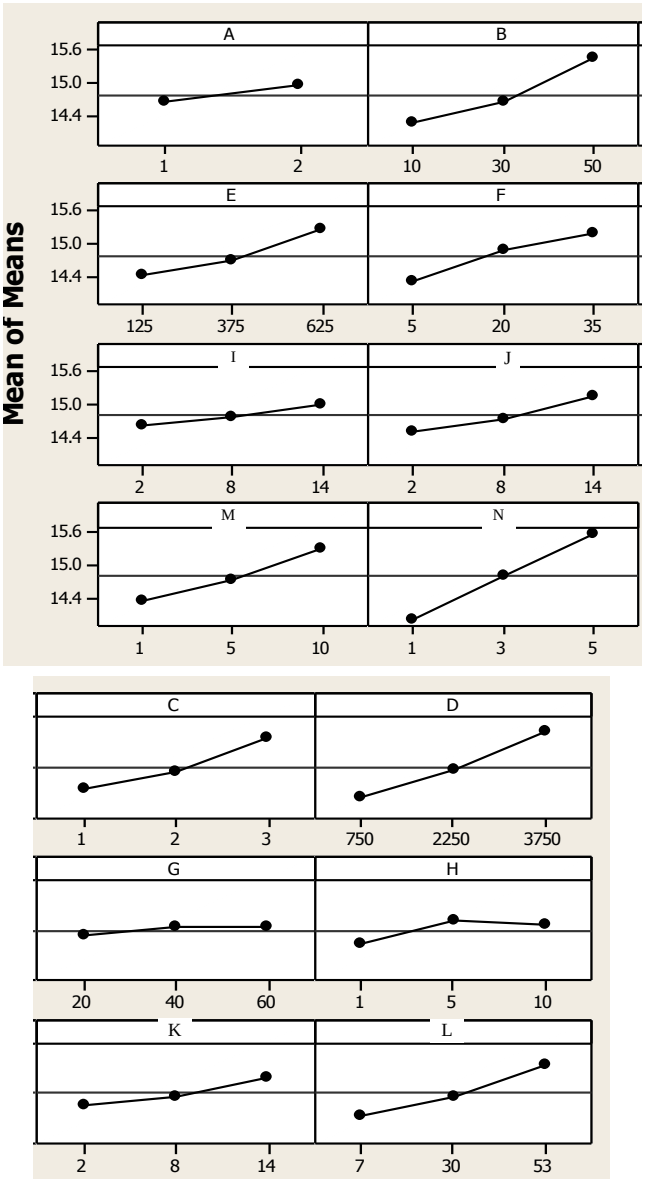

Fig. 2. Main effects plot.

However, the same may not be true beyond the present range of parameters and more so when there is a non-linear relationship between the effort drivers and response. Hence, the neural network based approach has also been attempted as an alternate method and a comparison is made between the two approaches.

5. NEURAL NETWORK MODELING

ANN is a class of flexible non-linear model inspired by the way in which the human brain processes information. Given an appropriate number of hidden layer units, it is well established that ANN can approximate any non-linear function, to a reasonable degree of accuracy [23]. The flexibility and generalization ability of ANN have made them a popular modeling tool across different research areas in recent years. ANN trained using an algorithm learns stagewise, progressing from fairly simple to more complex mapping functions. The mean-square error decreases with an increasing number of iterations during training.

The NN architecture chosen in our case was the 3 layer back-propagation, with 14 inputs, 14 hidden neurons and 1 output (14-11-1), as shown in Figure 3. The uni-modal sigmoid activation function in hidden layer and output layer was used in the present study. We initially kept only one hidden layer with hidden nodes equal to the inputs i.e. 14. The number of hidden nodes was gradually increased from 14 and the reduction in SSE observed. During trials, the minimum MSE did not change significantly with increased hidden nodes. Hence, a simplified NN architecture with only one hidden layer and minimum number of hidden neurons was finalized. The network was trained using 27 samples (50% of input data set) and rest 25% each were used for validation and testing. In this work, we have used the Matlab NN toolbox functions ([24], [25]). This toolbox provides utility functions for creating and training NNs, and verification and validation of NNs by simulation and visualization.

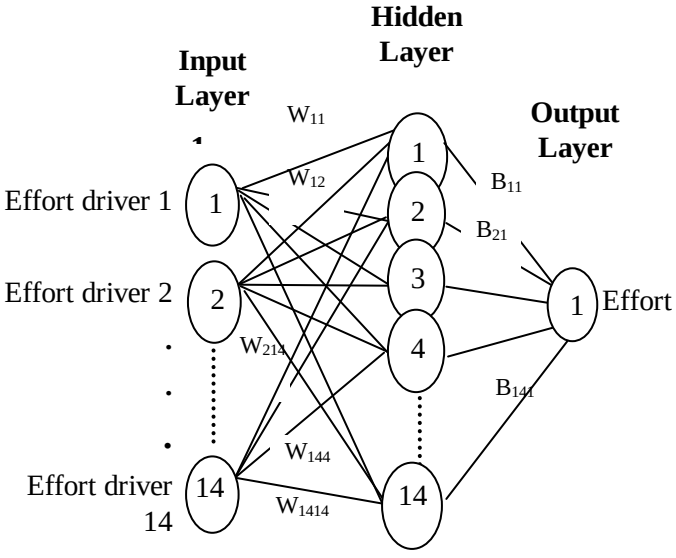

Fig. 3. NN architecture.

6. ANALYSIS AND VALIDATION OF RESULTS

A comparison of the 14-14-1 NN output with measured experimental values of effort shows the % error varying from +4.32 to -38.56, +18.72 to -5.87 and +6.12 to -2.31 for the training dataset (18 nos.), testing dataset (9 nos.) and validation dataset (9 nos.), respectively. The average % error though is significantly small at -1.93, 2.40 and 0.46, respectively. The next step was to perform analysis of the network response. The results of training of available data with a 14-14-1 architecture are shown in Figure 4. The obtained trends were as expected since the test set error and the validation set error have similar characteristics and tend to converge very fast (40 epochs). Further, any significant overfitting does not seem to have occurred. The sum of squared errors SSE for training (10.31), testing (0.99) and validation (0.64) as against a target of 0.0 are on expected lines and similar to that given in literature.

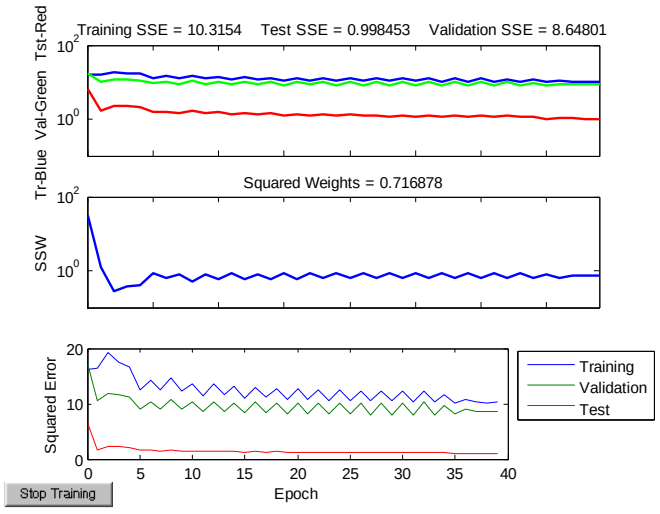

Fig. 4. NN simulation plot.

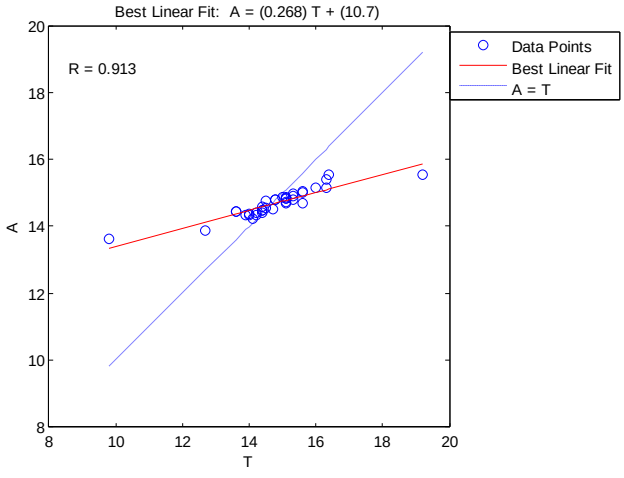

Fig. 5. Regression plot showing the target and actual values as predicted by NN.

Linear regression analysis between the network outputs and the corresponding targets was performed as shown in Fig 5. The two outliers of smallest effort (9.8) and largest effort (19.2) show larger errors of -38.56% and 18.72%. Hence, a log transformation (i.e. log(inputs) and log(output effort) has also been attempted, significantly reducing the above errors to -13.98% and 7.28%. For all the above details reference may be made to Appendix 2. A much simplified NN architecture was able to effectively and successfully model the highly non-linear relationship between the 14 variables and a single output parameter, as is evident from the high correlation coefficient 'R' value (around 0.9), for multiple runs of the code (Fig. 5).

The predicted effort (based on uncoded inputs / actual values) using the regression equation (Eq. 1) has been shown in the last column of Appendix 1. It can be inferred from the predicted values that the Taguchi approach based predicted effort models the effort with high accuracy validating the proposed approach. However, a single model will be insufficient to deal with vastly varying nature of projects.

7. CONCLUSIONS

In this paper, effectiveness of NN modeling approach of effort estimation for outsourced software maintenance projects was presented. The NN model trained using experimental data was found to have good generalization capabilities and is able to successfully predict the effort closely matching the experimental observations. Since the effect of various cost drivers on effort is often quite complex, ANN can be used as an effective tool to model and predict the SM effort. However, the models should also be evaluated by exploring the model sensitivity and scalability on a variety of historical and unseen input data [26].

8. REFERENCES

[1] IEEE Standard 1219: 1998. Standard for software maintenance, IEEE Computer Society Press.

[2] Boehm, B., Abts, C. and Chulani, S. 2000. Software development cost estimation approaches – a survey, Ann. Software Eng., 10, 177–205.

[3] Shukla, R and Misra, A. K. 2009. AI Based Framework for Dynamic Modeling of Software Maintenance Effort Estimation, Proceedings of International Conference on Computer and Automation Engineering, 313-317.

[4] Rao, B. S. and Sarda, N. L. 2003. Effort drivers in maintenance outsourcing - an experiment using Taguchi's methodology, Proceedings of Seventh IEEE European Conference on Software Maintenance and Reengineering, 1-10.

[5] Ahn, Y., Suh, J., Kim, S. and Kim, H. 2003. The software maintenance project effort estimation model based on function points, J. Software Maint. and Evol.: Res. and Practice, 15, 2, 71-78.

[6] Tronto, I. F. B., Silva, J. D. S. and Anna, N. S. 2008. An investigation of artificial neural networks based prediction systems in software project management, J. Syst. Software, 81, 356-367.

[7] Martín, C. L., Márquez, C. Y. and Tornés, A. G. 2008. Predictive accuracy comparison of fuzzy models for software development effort of small programs, J. Syst. Software, 81, 6, 949-960.

[8] Park, H. and Baek, S. 2008. An empirical validation of a neural network model for software effort estimation, Exp. Syst. Applic., 35, 3, 929-937.
[9] Jorgensen, M. 2004. A review of studies on expert estimation of software development effort, J. Syst. Software, 70, 1-2, 37-60.
[10] Jorgensen, M. 1995. Experience with accuracy of software maintenance task effort prediction models, IEEE Trans. Software Eng., 674-681.
[11] Grimstad, S. and Jørgensen, M. 2007. Inconsistency of expert judgment-based estimates of software development effort, J. Syst. Software, 80, 11, 1770-1777.
[12] Bhatt, P., Shroff, G., Anantram, C. and Misra, A. K. 2006. An nfluence model for factors in outsourced software maintenance, J. Software Maint. and Evol.: Res. and Practice, 18, 385-423.
[13] Shukla, R. and Misra, A. K. 2008. Estimating software maintenance effort - A neural network approach, Proceedings of the 1st India Software Engineering Conference - ISEC, Hyderabad, India, 107-112.
[14] Khoshgoftaar, T. M. I. and Abran, A. 2002. Can neural networks be easily interpreted in software cost estimation, IEEE Trans. Software Eng., 1162-1167.
[15] Pendharkar, P. C., Subramanian, G. H. and Rodger, J. A. 2005. A probabilistic model for predicting software development effort, IEEE Trans. Software Eng., 31, 7, 615-624.
[16] Witting, G. and. Finnie, G. 1994. Using artificial neural networks and function points to estimate 4GL software development effort. J. Inform. Systems, 1, 2, 87–94.
[17] Srinivazan, K. and Fisher, D. 1995. Machine learning approaches to estimating software development effort. IEEE Trans. Software Eng., 21, 2, 126–137.
[18] Boetticher, G. D. 2001. An assessment of metric contribution in the construction of a neural network-based effort estimator, Proceedings of Second Int. Workshop on Soft Computing Applied to Software Engineering.
[19] Shukla, K. K. 2000. Neuro-genetic prediction of software development effort, Inform. Software Tech., 42, 701-713.
[20] Huang, S. J., Chiu, N. H. and Chen, L. W. 2008. Integration of the grey relational analysis with genetic algorithm for software effort estimation, European J. Oper. Research, 188, 3, 898-909.
[21] Phadke, M. S. 1989. Quality Engineering Using Robust Design, Eaglewood cliffs, NJ: Prentice Hall.
[22] www.minitab.com
[23] Haykin, S. 1999. Neural networks: A comprehensive foundation, Prentice Hall.
[24] Shukla, M. and Tambe, P. B. 2010. Predictive modeling of surface roughness and kerf widths in abrasive water jet cutting of kevlar composites using artificial neural network, Int. J. Mach. Machin. of Mater., In press.
[25] www.mathworks.com
[26] Aggarwal, K. K., Singh, Y., Chandra, P. and Puri, M. 2004. Sensitivity analysis of fuzzy and neural network models, ACM SIGSOFT Software Eng. Notes, 29, 5, 1-5.

Appendix 1

Sl. No	A	B	C	D	E	F	G	H	J	K	L	M	N	O	Effort_Ave	Predicted Effort
1.	1	10	1	750	125	5	20	1	2	2	2	7	1	1	9.8	9.5528
2.	1	10	2	2250	375	20	40	5	8	8	8	30	5	3	14.1	13.8528
3.	1	10	3	3750	625	35	60	10	14	14	14	53	10	5	19.2	18.9528
4.	1	10	1	750	125	5	40	5	8	8	14	53	10	5	14.5	14.5194
5.	1	10	2	2250	375	20	60	10	14	14	2	7	1	1	12.7	12.7194
6.	1	10	3	3750	625	35	20	1	2	2	8	30	5	3	15.1	15.1194
7.	2	10	1	750	375	35	20	5	14	14	2	30	5	5	14.7	14.7639
8.	2	10	2	2250	625	5	40	10	2	2	8	53	10	1	14.0	14.0639
9.	2	10	3	3750	125	20	60	1	8	8	14	7	1	3	14.4	14.4639
10.	2	10	1	750	625	20	20	10	8	14	8	7	10	3	14.2	14.3639
11.	2	10	2	2250	125	35	40	1	14	2	14	30	1	5	14.4	14.5639
12.	2	10	3	3750	375	5	60	5	2	8	2	53	5	1	14.2	14.3639
13.	1	30	1	2250	625	5	60	5	2	14	14	30	1	3	13.9	14.1944
14.	1	30	2	3750	125	20	20	10	8	2	2	53	5	5	15.1	15.3944
15.	1	30	3	750	375	35	40	1	14	8	8	7	10	1	13.6	13.8944
16.	1	30	1	2250	625	20	20	1	14	8	14	53	5	1	14.0	14.2611
17.	1	30	2	3750	125	35	40	5	2	14	2	7	10	3	15.1	15.3611
18.	1	30	3	750	375	5	60	10	8	2	8	30	1	5	13.6	13.8611
19.	2	30	1	2250	125	35	60	10	2	8	8	7	5	5	14.8	14.4722
20.	2	30	2	3750	375	5	20	1	8	14	14	30	10	1	15.0	14.6722
21.	2	30	3	750	625	20	40	5	14	2	2	53	1	3	15.6	15.2722
22.	2	30	1	2250	375	35	60	1	8	2	2	53	10	3	15.1	14.8722
23.	2	30	2	3750	625	5	20	5	14	8	8	7	1	5	15.6	15.3722
24.	2	30	3	750	125	20	40	10	2	14	14	30	5	1	14.4	14.1722
25.	1	50	1	3750	375	5	40	10	14	2	14	7	5	3	15.1	14.8194
26.	1	50	2	750	625	20	60	1	2	8	2	30	10	5	16.0	15.7194
27.	1	50	3	2250	125	35	20	5	8	14	8	53	1	1	15.6	15.3194
28.	1	50	1	3750	375	20	40	1	2	14	8	53	1	5	16.3	16.2528
29.	1	50	2	750	625	35	60	5	8	2	14	7	5	1	14.4	14.3528
30.	1	50	3	2250	125	5	20	10	14	8	2	30	10	3	15.3	15.2528
31.	2	50	1	3750	625	35	40	10	8	8	2	30	1	1	14.8	15.1639
32.	2	50	2	750	125	5	60	1	14	14	8	53	5	3	14.5	14.8639
33.	2	50	3	2250	375	20	20	5	2	2	14	7	10	5	16.4	16.7639
34.	2	50	1	3750	125	20	60	5	14	2	8	30	10	1	15.3	15.2639
35.	2	50	2	750	375	35	20	10	2	8	14	53	1	3	15.3	15.2639
36.	2	50	3	2250	625	5	40	1	8	14	2	7	5	5	16.3	16.2639

Appendix 2

Sl. No.	Actual Effort	NN Predicted Effort	% error	% error (log transformation)
TRAINING SET				
1	9.8	13.61	**-38.56**	-13.98
2	14.1	14.24	-0.70	-1.10
5	12.7	13.88	-8.59	-3.19
6	15.1	14.72	1.85	0.84
9	14.4	14.49	-0.51	-0.11
10	14.2	14.41	-1.25	-0.48
13	13.9	14.35	-2.95	-1.41
14	15.1	14.85	0.74	0.58
17	15.1	14.82	1.29	0.60
18	13.6	14.45	-6.34	-2.20
21	15.6	14.99	3.41	1.24
22	15.1	14.82	1.46	0.15
25	15.1	14.71	2.28	0.92
26	16.0	15.16	4.32	1.93
29	14.4	14.48	-0.48	-0.05
30	15.3	14.91	1.84	1.21
33	16.4	15.53	4.32	1.87
34	15.3	14.79	3.12	1.42
		Average Error	-1.93	-0.65
TESTING SET				
3	19.2	15.55	**18.72**	7.28
7	14.7	14.52	1.14	0.68
11	14.4	14.59	-1.50	-0.25
15	13.6	14.43	-5.87	-2.12
19	14.8	14.79	-0.40	-0.54
23	15.6	15.05	2.74	1.47
27	15.6	14.68	5.63	2.93
31	14.8	14.80	-0.29	0.55
35	15.3	14.97	1.47	0.84
		Average Error	+2.40	+1.20
VALIDATION SET				
4	14.5	14.53	-0.18	0.31
8	14.0	14.33	-2.08	-0.63
12	14.2	14.34	-0.73	-0.08
16	14.0	14.36	-2.31	-1.07
20	15.0	14.87	0.54	0.47
24	14.4	14.41	0.21	-0.06
28	16.3	15.16	6.12	2.82
32	14.5	14.78	-2.01	-0.78
36	16.3	15.40	4.64	2.12
		Average Error	+0.46	+0.34