An Ensemble of Computational Intelligence Models for Software Maintenance Effort Prediction

Hamoud Aljamaan, Mahmoud O. Elish, and Irfan Ahmad

Information & Computer Science Department
King Fahd University of Petroleum & Minerals
Dhahran, Saudi Arabia
{hjamaan,elish,irfanics}@kfupm.edu.sa

Abstract. More accurate prediction of software maintenance effort contributes to better management and control of software maintenance. Several research studies have recently investigated the use of computational intelligence models for software maintainability prediction. The performance of these models however may vary from dataset to dataset. Consequently, computational intelligence ensemble techniques have become increasingly popular as they take advantage of the capabilities of their constituent models toward a dataset to come up with more accurate or at least competitive prediction accuracy compared to individual models. This paper proposes and empirically evaluates an ensemble of computational intelligence models for predicting software maintenance effort. The results confirm that the proposed ensemble technique provides more accurate prediction compared to individual models, and thus it is more reliable.

Keywords: Computational intelligence, Ensemble techniques, Software maintenance, Prediction.

1 Introduction

Software maintenance is one of the most difficult and costly tasks in the software development lifecycle [24, 40]. Accurate prediction of software maintainability can be useful to support and guide [9]: software related decision making; maintenance process efficiency; comparing productivity and costs among different projects; resource and staff allocation, and so on. As a result, future maintenance effort can be kept under control.

Recent research studies have investigated the use of computational intelligence models for software maintainability prediction [10, 22, 40]. These models have different prediction capabilities and none of them has proved to be the best under all conditions. Performance of these models may vary from dataset to dataset. Computational intelligence ensemble techniques take advantage of the capabilities of their constituent models toward a dataset to come up with more accurate or at least competitive prediction accuracy compared to individual models. They have high potential in providing reliable predictions.

I. Rojas, G. Joya, and J. Cabestany (Eds.): IWANN 2013, Part I, LNCS 7902, pp. 592–603, 2013.

This paper proposes and empirically evaluates an ensemble technique of computational intelligence models for predicting software maintenance effort. The rest of this paper is organized as follows. Section 2 reviews related work. In Section 3, we describe the proposed computational intelligence ensemble technique. In Section 4, we describe the ensemble constituent models. In Section 5, we present the discussions on the conducted empirical evaluation and its results. In Section 6, we present the conclusions and suggest directions for future work.

2 Related Work

Several research studies have investigated the relationship between object-oriented metrics and the maintainability of object-oriented software systems, and they found significant correlations between them [2, 6, 12, 24, 26]. These metrics can thus be used as good predictors of software maintainability. Furthermore, recent research studies have investigated the use of computational intelligence models for software maintainability prediction. These models were constructed using object-oriented metrics as input variables. Such models include TreeNet [11], multivariate adaptive regression splines [40], naïve bayes [22], artificial neural network [35, 40], regression tree [22, 40], and support vector regression [40].

Thwin and Quah [35] predicted the software maintainability as the number of lines changed per class. Their experimental results found that General Regression neural network predict maintainability more accurately than Ward network model. Koten and Gray [22] evaluated and compared the naïve bayes classifier with commonly used regression-based models. Their results suggest that the naïve bayes model can predict maintainability more accurately than the regression-based models for one system, and almost as accurately as the best regression-based model for the other system. Zhou and Leung [40] explored the employment of multiple adaptive regression splines (MARS) in building software maintainability prediction models. MARS was evaluated and compared against multivariate linear regression models, artificial neural network models, regression tree models, and support vector models. Their results suggest that, for one system, MARS can predict maintainability more accurately than the other four typical modeling techniques. Then, Elish and Elish [11] extended the work done by Zhou and Leung [40] to investigate the capability of TreeNet technique in software maintainability prediction. Their results indicate that TreeNet can yield improved, or at least competitive, prediction accuracy over previous maintainability prediction models.

Recently, computational intelligence ensemble models have received much attention and have demonstrated promising capabilities in improving the accuracy over single models [4, 34]. Ensemble models have been used in the area of software engineering prediction problems. They have been used in software reliability prediction [39], software project effort estimation [4], and software fault prediction [1, 19]. In addition, they have been used in many real applications such as face recognition [14, 18], OCR [25], seismic signal classification [33] and protein structural class prediction [3]. However, according to the best knowledge of the authors, none of the

computational intelligence ensemble techniques have been used in the area of software maintenance effort prediction.

3 The Ensemble Technique

An ensemble of computational intelligence models uses the outputs of all its individual constituent prediction models, each being assigned a certain priority level, and provide the final output with the help of an arbitrator [29]. There are single-model ensembles and multi-model ensembles. In single-model ensembles, the individual constituent prediction models are of the same type (for example, all of them could be radial basis function network), but each with randomly generated training set. Examples of single-model ensembles include Bagging [5] and Boosting [13]. In multi-model ensembles, there are different individual constituent prediction models. This study focuses on multi-model ensembles.

The multi-model ensembles can be further classified, according to the design of the arbitrator, into linear ensembles and nonlinear ensembles [20]. In linear ensembles, the arbitrator combines the outputs of the constituent models in a linear fashion such as average, weighted average, etc. In nonlinear ensembles, no assumptions are made about the input that is given to the ensemble [20]. The output of the individual prediction models are fed into an arbitrator, which is a nonlinear prediction model such as neural network which when trained, assigns the weights accordingly. In this study, we propose a linear computational intelligence ensemble technique, which is described next.

The proposed ensemble takes the advantage of the fact that individual prediction models have different errors across the used dataset partitions. The idea behind this ensemble is that across the dataset partitions, take the best model in training based upon a certain criterion in that partition. In this study, the criterion is mean magnitude of relative error (MMRE). Fig. 1 provides a formal description of the ensemble.

```
Choose dataset with N observations
Choose M individual prediction models
Set K for K folds cross validation
For each k ∈ K fold
    For each m ∈ M model
        Apply model m on the training set for fold(k)
        Calculate training error E, based on a certain criterion
        Store error E
    End for
    Select the best model b ∈ M, based on training error E
    For each n ∈ N observation in the testing set for fold(k)
        EnsembleOutput = the result of applying model b on observation n
    End for
End for
```

Fig. 1. The ensemble technique

4 Ensemble Constituent Models

In this section we briefly describe the individual computational intelligence models that are used as base for the computational intelligence ensemble technique, i.e. the ensemble constituent models. These models were chosen because they are commonly and widely used in the literature of software quality and effort prediction. These models were built using WEKA machine learning toolkit [38], and their parameters were initialized using the default values.

4.1 Multilayer Perceptron

Multilayer Perceptron (MLP) [17] are feedforward networks that consist of an input layer, one or more hidden layers of nonlinearly activating nodes and an output layer. Each node in one layer connects with a certain weight to every other node in the following layer. MLP uses backpropagation algorithm as the standard learning algorithm for any supervised-learning.

The parameters of this model were initialized as follows. Backpropagation algorithm was used for training. Sigmoid was used as an activation function. Number of hidden layers was 5. Learning rate was 0.3 with momentum 0.2. Network was set to reset with a lower learning rate. Number of epochs to train through was 500. Validation threshold was 20.

4.2 Radial Basis Function Network

Radial Basis Function Network (RBF) [30] is an artificial neural network that uses radial basis functions as activation functions to provide a flexible way to generalize linear regression function. Commonly used types of radial basis functions include Gaussian, Multiquadric, and Polyharmonic spline. RBF models with Gaussian basis functions possess desirable mathematical properties of universal approximation and best approximation. A typical RBF model consists of three layers: an input layer, a hidden layer with a non-linear RBF activation function and a linear output layer.

The parameters of this model were initialized as follows. A normalized Gaussian radial basis function network was used. Random seed to pass on to K-means clustering algorithm was 1. Number of clusters for K-means clustering algorithm to generate was 2, with minimum standard deviation for clusters set to 0.1.

4.3 Support Vector Machines

Support Vector Machines (SVMs) was proposed by Vapnik [36] based on the structured risk minimization (SRM) principle. SVMs are a group of supervised learning methods that can be applied to classification or regression problems. SVMs aim to minimize the empirical error and maximize the geometric margin. SVM model is defined by these parameters: complexity parameter C, extent to which deviations are tolerated ε, and kernel.

The parameters of this model were initialized as follows. The cost parameter C was set to 1, with polynomial as SVMreg kernel. The most popular (RegSMOImproved) algorithm [32] was used for parameter learning.

4.4 M5 Model Tree

M5 Model tree (M5P) [31, 38] is an algorithm for generating M5 model trees that predicts numeric values for a given instance. To build a model tree, the M5 algorithm starts with a set of training instances. The tree is built using a divide-and-conquer method. At a node, starting with the root node, the instance set that reaches it is either associated with a leaf or a test condition is chosen that splits the instances into subsets based on the test outcome. In M5, the test that maximizes the error reduction is used. Once the tree has been built, a linear model is constructed at each node. The linear model is a regression equation.

The parameters of this model were initialized as follows. M5 algorithm was used for generating M5 model trees [31, 37]. Pruned M5 model trees were built, with 4 instances as the minimum number of instances allowed at a leaf node.

5 Empirical Evaluation

This empirical evaluation aims to determine the extent to which the proposed ensemble technique offers an increase in software maintenance effort prediction accuracy over individual models.

5.1 Datasets

We used two popular object-oriented software maintainability datasets published by Li and Henry [24]: UIMS and QUES datasets. These datasets are publicly available which makes our study verifiable, repeatable, and reputable [9]. The UIMS dataset contains class-level metrics data collected from 39 classes of a user interface management system, whereas the QUES dataset contains the same metrics collected from 71 classes of a quality evaluation system. Both systems were implemented in Ada. Both datasets consist of eleven class-level metrics: ten independent variables and one dependent variable.

The independent (input) variables are five Chidambar and Kemerer metrics [7]: WMC, DIT, NOC, RFC, and LCOM; four Li and Henry metrics [24]: MPC, DAC, NOM, SIZE2; and one traditional lines of code metric (SIZE1). Table 1 provides brief description for each metric.

The dependent (output) variable is a maintenance effort proxy measure, which is the actual number of lines in the code that were changed per class during a 3-year maintenance period. A line change could be an addition or a deletion. A change in the content of a line is counted as a deletion and an addition [24].

Table 1. Independent variables in the datasets

Metric	Description
WMC	Count of methods implemented within a class
DIT	Level for a class within its class hierarchy
NOC	Number of immediate subclasses of a class
RFC	Count of methods implemented within a class plus the number of methods accessible to an object class due to inheritance
LCOM	The average percentage of methods in a class using each data field in the class subtracted from 100%
MPC	The number of messages sent out from a class
DAC	The number of instances of another class declared within a class
NOM	The number of methods in a class
SIZE1	The number of lines of code excluding comments
SIZE2	The total count of the number of data attributes and the number of local methods in a class

Previous studies [10, 22, 40], on both datasets, indicate that both datasets have different characteristics, and therefore, considered heterogeneous and a separate maintenance effort prediction model is built for each dataset.

5.2 Accuracy Evaluation Measures

We used de facto standard and commonly used accuracy evaluation measures that are based on magnitude of relative error (MRE) [8]. These measures are mean magnitude of relative error (MMRE), standard deviation magnitude of relative error (StdMRE), and prediction at level q (Pred(q)). MMRE over a dataset of n observations is calculated as follows:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i$$

where MREi is a normalized measure of the discrepancy between the actual value (x_i) and the predicated value ($\hat{x}_i$) of observation i. It is calculated as follows:

$$MRE_i = \frac{\left|x_i - \hat{x}_i\right|}{x_i}$$

In addition to MMRE, we used StdMRE since it is less sensitive to the extreme values compared to MMRE. Pred(q) is a measure of the percentage of observations whose MRE is less than or equal to q. It is calculated as follows:

$$\Pr ed(q) = \frac{k}{n}$$

where k is the number of observations whose MRE is less than or equal to a specified level q, and n is the total number of observations in the dataset. An acceptable value for level q is 0.3, as indicated in the literature [8, 22, 40]. We therefore adopted that value.

5.3 Results and Analysis

We used a 10-fold cross validation [21] (i.e. k-fold cross validation, with k set to 10), which is a common validation technique used to evaluate the performance of prediction models. In 10-fold cross validation; a dataset is randomly partitioned into 10 folds of equal size. For 10 times, 9 folds are picked to train the models and the remaining fold is used to test them, each time leaving out a different fold.

Table 2 provides the results obtained from applying the individual computational intelligence models on UIMS dataset, as well as the results achieved by the ensemble model. Among the individual models, the MLP model achieved the best result in general, whereas the RBF model was the worst. It can be observed that the ensemble model outperformed all the individual models.

Table 2. Prediction accuracy results: UIMS dataset

	Individual Models				Ensemble Model
	MLP	**RBF**	**SVM**	**M5P**	
MMRE	1.39	3.23	1.64	1.67	**0.97**
StdMRE	2.40	4.43	2.38	2.75	**1.61**
Pred(0.3)	23.33	15	20	23.33	**25**

Fig. 2 shows the box plot of MRE values for each model, where the middle of each box represents the MMRE for each model. As can be seen, the ensemble model has the narrowest box and the smallest whiskers (i.e. the lines above and below from the box). Moreover, its box and whiskers are lower than those of the individual models, which clearly indicate that the ensemble model outperforms the individual models. Fig. 3 shows a histogram of the achieved Pred(0.30) value by each model. From the figure it can be seen clearly that the ensemble model achieved the best Pred(0.30).

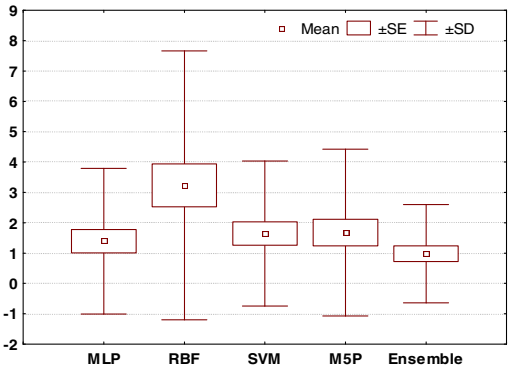

Fig. 2. Box plots of MRE for each model: UIMS dataset

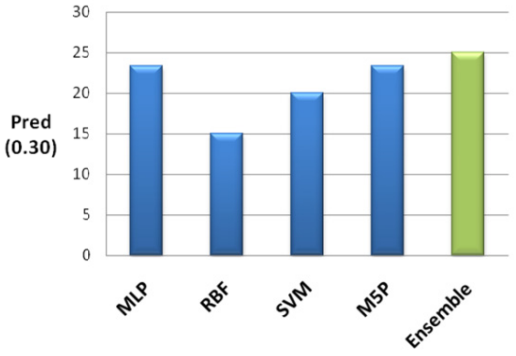

Fig. 3. Pred(0.30) for each model: UIMS dataset

Table 3 provides the results obtained from applying the individual computational intelligence models on QUES dataset, as well as the results achieved by the ensemble model. Among the individual models, the SVM model achieved the best result, whereas the RBF model was the worst. It also can be observed that the ensemble model outperformed all the individual models.

Table 3. Prediction accuracy results: QUES dataset

	Individual Models				Ensemble Model
	MLP	**RBF**	**SVM**	**M5P**	
MMRE	0.71	0.96	0.44	0.54	**0.41**
StdMRE	0.65	1.52	0.39	0.56	**0.32**
Pred(0.3)	40	36.66	56.66	51.66	**60**

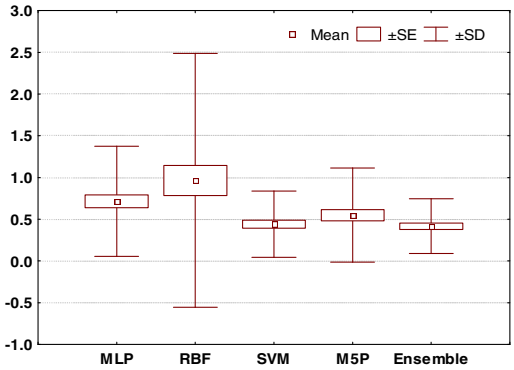

Fig. 4. Box plots of MRE for each model: QUES dataset

Fig. 4 shows the box plot of MRE values for each model, where the middle of each box represents the MMRE for each model. It can be observed that the ensemble model has the narrowest box and the smallest whiskers. Its box and whiskers are also

lower than those of the individual models, which clearly indicate that the ensemble model outperforms the individual models in this dataset too. Fig. 5 shows a histogram of the achieved Pred(0.30) value by each model. The ensemble model achieved the highest Pred(0.30) value, i.e. 60%.

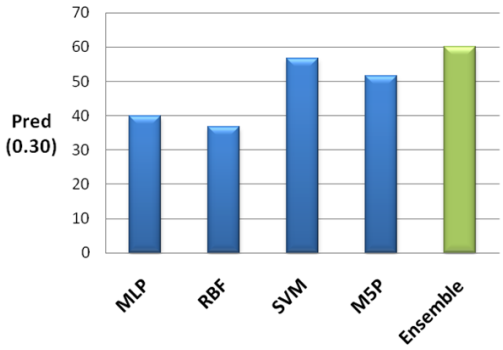

Fig. 5. Pred(0.30) for each model: QUES dataset

When considering the results from both datasets, there are two main interesting observations. First, the results support that the performance of the individual prediction models may vary from dataset to dataset; the MLP model was the best in the UIMS dataset while the SVM model was the best in the QUES dataset. Second, the ensemble model outperformed the individual models in both datasets.

6 Conclusions and Future Work

In this paper we presented an ensemble of computational intelligence models for predicting software maintenance effort. As ensemble constituent models, four popular prediction models (MLP, RBF, SVM, and M5P) were used. The prediction accuracy of the ensemble technique was empirically evaluated using two public object-oriented software maintainability datasets. The results indicate that ensemble technique provides more accurate prediction compared to individual models, and thus it is more reliable. It is worth noting that ensemble technique will be in general more complex and computationally expensive as compared to using a single prediction model but the benefits of using the ensemble in terms of prediction accuracy and robustness outweighs this penalty.

There are possible directions for future work, which include: investigating nonlinear ensemble models and comparing their performance with linear ensemble models; considering other ensemble constituent models; applying computational intelligence ensemble techniques to other software engineering prediction problems such as fault prediction. Both theoretical [15, 23] and empirical [16, 27, 28] research studies have demonstrated that a good ensemble is one where the individual prediction models in the ensemble are both accurate and make their errors on different parts of the input space. Therefore, one important direction of future work is to investigate different sets of ensemble constituent models.

Acknowledgements. The authors wish to acknowledge King Fahd University of Petroleum and Minerals (KFUPM) for utilizing the various facilities in carrying out this research.

References

1. Aljamaan, H., Elish, M.: An Empirical Study of Bagging and Boosting Ensembles for Identifying Faulty Classes in Object-Oriented Software. In: IEEE Symposium on Computational Intelligence and Data Mining, pp. 187–194 (2009)
2. Bandi, R.K., Vaishnavi, V.K., Turk, D.E.: Predicting Maintenance Performance Using Object-Oriented Design Complexity Metrics. IEEE Transactions on Software Engineering 29(1), 77–87 (2003)
3. Bittencourt, V.G., Abreu, M.C.C., Souto, M.C.P.D., Canuto, A.M.D.P.: An empirical comparison of individual machine learning techniques and ensemble approaches in protein structural class prediction. In: International Joint Conference on Neural Networks, pp. 527–531 (2005)
4. Braga, P.L., Oliveira, A.L.I., Ribeiro, G.H.T., Meira, S.R.L.: Bagging Predictors for Estimation of Software Project Effort. In: International Joint Conference on Neural Networks, pp. 1595–1600 (2007)
5. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
6. Briand, L.C., Bunse, C., Daly, J.W.: A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs. IEEE Transactions on Software Engineering 27(6), 513–530 (2001)
7. Chidamber, S.R., Kemerer, C.F.: A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering 20(6), 476–493 (1994)
8. Conte, S., Dunsmore, H., Shen, V.: Software Engineering Metrics and Models. Benjamin/Cummings, Menlo Park (1986)
9. De Lucia, A., Pompella, E., Stefanucci, S.: Assessing effort estimation models for corrective maintenance through empirical studies. Information and Software Technology 47(1), 3–15 (2005)
10. Elish, M., Elish, K.: Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study. In: 13th European Conference on Software Maintenance and Reengineering (CSMR 2009), pp. 69–78 (2009)
11. Elish, M.O., Elish, K.O.: Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study. In: 13th European Conference on Software Maintenance and Reengineering (CSMR 2009), pp. 69–78 (2009)
12. Fioravanti, F., Nesi, P.: Estimation and prediction metrics for adaptive maintenance effort of object-oriented systems. IEEE Transactions on Software Engineering 27(12), 1062–1084 (2001)
13. Freund, Y.: Boosting a weak learning algorithm by majority. Information and Computation 121(2), 256–285 (1995)
14. Gutta, S., Wechsler, H.: Face Recognition Using Hybrid Classifier Systems. In: IEEE International Conference on Neural Networks, pp. 1017–1022 (1996)
15. Hansen, L., Salamon, P.: Neural Network Ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence 12(10), 993–1001 (1990)

16. Hashem, S., Schmeiser, B., Yih, Y.: Optimal linear combinations of neural networks. Neural Networks 3, 1507–1512 (1994)
17. Haykin, S.: Neural Networks: A Comprehensive Foundation New Jersey. Prentice Hall, New Jersey (1999)
18. Huang, F.J., Zhou, Z., Zhang, H.-J., Chen, T.: Pose invariant face recognition. In: Proc. 4th IEEE Int. Conf. on Automatic Face and Gesture Recognition, France, pp. 245–250 (2000)
19. Khoshgoftaar, T.M., Geleyn, E., Nguyen, L.: Empirical Case Studies of Combining Software Quality Classification Models. In: Third International Conference on Quality Software, p. 40 (2003)
20. Kiran, N., Ravi, V.: Software reliability prediction by soft computing techniques. Journal of Systems and Software 81(4), 576–583 (2008)
21. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI), pp. 1137–1143 (1995)
22. Koten, C., Gray, A.: An application of Bayesian network for predicting object-oriented software maintainability. Information and Software Technology 48(1), 59–67 (2006)
23. Krogh, A., Vedelsby, J.: Neural Network Ensembles, Cross Validation, and Active Learning. In: Advances in Neural Information Processing Systems, vol. 7, pp. 231–238 (1995)
24. Li, W., Henry, S.: Object-Oriented Metrics that Predict Maintainability. Journal of Systems and Software 23(2), 111–122 (1993)
25. Mao, J.: A case study on bagging, boosting and basic ensembles of neural networks for OCR. In: Proc. IEEE Int. Joint Conf. on Neural Networks, pp. 1828–1833 (1998)
26. Misra, S.C.: Modeling Design/Coding Factors That Drive Maintainability of Software Systems. Software Quality Control 13(3), 297–320 (2005)
27. Opitz, D.W., Shavlik, J.W.: Actively searching for an effective neural-network ensemble. Connection Science 8(3/4), 337–353 (1996)
28. Opitz, D.W., Shavlik, J.W.: Generating Accurate and Diverse Members of a Neural-Network Ensemble. In: Advances in Neural Information Processing Systems, vol. 8, pp. 535–541 (1996)
29. Optiz, D., Maclin, R.: Popular Ensemble Methods: An Empirical Study. Journal of Artificial Intelligence Reseach 11, 169–198 (1999)
30. Poggio, T., Girosi, F.: Networks for approximation and learning. Proceedings of the IEEE 78(9), 1481–1497 (1990)
31. Quinlan, R.J.: Learning with Continuous Classes. In: 5th Australian Joint Conference on Artificial Intelligence, Singapore, pp. 343–348 (1992)
32. Shevade, S.K., Keerthi, S.S., Bhattacharyya, C., Murthy, K.R.K.: Improvements to the SMO Algorithm for SVM Regression. IEEE Transactions on Neural Networks 11(5), 1188–1193 (2000)
33. Shimshoni, Y., Intrator, N.: Classification of seismic signals by integrating ensembles of neural networks. IEEE Transactions on Signal Processing 46(5), 1194–1201 (1998)
34. Sollich, P.: Learning with Ensembles: How over-fitting can be useful. In: Advances in Neural Information Processing Systems, vol. 8, pp. 190–196 (1996)
35. Thwin, M.M.T., Quah, T.-S.: Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics. Journal of Systems and Software 76(2), 147–156 (2005)
36. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)

37. Wang, Y., Witten, I.H.: Induction of model trees for predicting continuous classes. In: Poster papers of the 9th European Conference on Machine Learning (1997)
38. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
39. Zheng, J.: Predicting software reliability with neural network ensembles. Expert Systems with Applications (2007)
40. Zhou, Y., Leung, H.: Predicting object-oriented software maintainability using multivariate adaptive regression splines. Journal of Systems and Software 80(8), 1349–1361 (2007)