Integrating uncertainty in software effort estimation using Bootstrap based Neural Networks

Safae Laqrichi, François Marmier, Didier Gourc, Jean Nevoux

Integrating uncertainty in software effort estimation using Bootstrap based Neural Networks

Safae Laqrichi * **Francois Marmier** * **Didier Gourc** *
Jean Nevoux **

* *University of Toulouse, Mines Albi, Industrial Engineering Center, Route de Teillet, Campus Jarlard, 81013 Albi Cedex 09, France (e-mail: firstname.lastname@ mines-albi.fr).*
** *ACAPI, 67 Avenue du Marchal Joffre, 92000 Nanterre, France (e-mail: jean.nevoux@acapi.fr).*

Abstract: Software effort estimation is a crucial task in the software project management. It is the basis for subsequent planning, control, and decision-making. Reliable effort estimation is difficult to achieve, especially because of the inherent uncertainty arising from the noise in the dataset used for model elaboration and from the model limitations. This research paper proposes a software effort estimation method that provides realistic effort estimates by taking into account uncertainty in the effort estimation process. To this end, an approach to introducing uncertainty in Neural Network based effort estimation model is presented. For this purpose, bootstrap resampling technique is deployed. The proposed method generates a probability distribution of effort estimates from which the Prediction Interval associated to a confidence level can be computed. This is considered to be a reasonable representation of reality, thus helping project managers to make well-founded decisions. The proposed method has been applied on a dataset from International Software Benchmarking Standards Group and has shown better results compared to traditional effort estimation based on linear regression.

Keywords: Uncertainty, Bootstrap, Prediction Interval, Neural network.

1. INTRODUCTION

Software effort estimation is the process of predicting the amount of effort required to develop the software project. It is an important activity in software project management as it provides the basis for subsequent planning, control, and decision making (Morgenshtern et al., 2007). Accurate effort estimation is important because a low cost estimate may result in financial loss or compromise the quality of the software developed, while a high cost can lead to misallocation of resources and a backlog of required software (Lee et al., 1998). Several surveys have reported that 60-80% of all software projects expend on average 30-40% more effort than originally estimated (Molokken and Jorgensen, 2003). This means that the average estimation error is high and that the estimates are unrealistic.

In recent decades, various effort estimation methods have been developed in order to improve the estimates accuracy of the effort estimates. However, there is no single model that completely satisfies the need for objective and realistic estimates in all circumstances (Basha and Ponnurangam, 2010). The majority of existing methods are elaborated from old historical data and for specific organizations, thus it is difficult to adapt them to new project contexts and environments. Therefore, in order to accurately estimate the effort, an organization requires estimation methods that are based on its own performance, working practices and software experience (Kitchenham and Linkman, 1997).

Estimates are guesses regarding future performance, based on available knowledge and estimation methods (Morgenshtern et al., 2007). Obviously, these guesses do not exactly mirror the actual outcome, due to the uncertainty present in the effort estimation process, especially in the early stages of software development. Uncertainty is inherently present in (1) the dataset, used in the elaboration of the estimation model, that can contain noisy and imprecise information, (2) the model that can not present the exact relationship between the effort and the influent variables. It is important to take account of uncertainty in the estimation process. Associating an uncertainty to effort estimates may provide the user with a sense of how accurate the estimates are likely to be and thus help him in making the right decisions (Jorgensen and Sjoeberg, 2003). Taking account of these observations, our study aims to elaborate an effort estimation method based on NN that can take into account the inherent uncertainty present in the effort estimation process. The provided effort estimates should be realistic and reflect the reality.

The remainder of this paper is structured as follows: Section 2 provides an insight into software effort estimation and uncertainty. Section 3 proposes an effort estimation method that takes account of uncertainty. Section 4 evaluates the proposed method on a dataset and discusses the results. The final section draws a conclusion summarizing the present study and outlining perspectives and future works.

2. LITERATURE REVIEW

2.1 Effort estimation methods

In the last decades, various effort estimation approaches have been developed. They can be classified in four main categories. (1) Expert judgment-based methods that draw upon the expert intuition and experience gained from previously executed projects, such as Delphi (Jorgensen, 2004). (2) Analogy based-methods that consist in identifying one or more historical projects that are similar to the project being developed and in using their known effort values to generate the estimated effort (Shepperd et al., 1996). (3) Parametric model-based methods, which rely mainly on equations elaborated using historical data and expressing the effort as a function of discriminant parameters influencing that effort, called effort drivers. Examples include COCOMO (Boehm, 2000). (4) Machine learning-based methods, which model the complex relationship between the effort and the effort drivers using artificial intelligence techniques like Neural Networks (NN) and fuzzy logic (Srinivasan and Fisher, 1995).

Most parametric effort estimation methods, such as COCOMO, have their difficulties and limitations. First, as they are elaborated from old historical data and for specific organizations, it is difficult to adapt them to new project contexts and environments. Moreover, they are not able to provide an effective model for the complex relationships between effort and effort drivers (Ahmed and Muzaffar, 2009). Effort estimation methods based on artificial intelligence techniques, especially NN, overcome most of these problems. NN effort estimation models are able to learn from previous data, to be adapted for any organization and project context, to be updated over time and to model complex relationships (Park and Baek, 2008; Idri et al., 2010; Setiono et al., 2010).

2.2 Neural Networks (NN)

NN is a massively-parallel adaptive network of simple nonlinear computing elements called neurons, which are intended to abstract and model some of the functionality of the human nervous system in an attempt to partially capture some of its computational strengths (Haykin, 1999).

There are a multitude of NN architecture and structure; the most used one is called Multilayer Perceptron (MLP) (see figure 1). It consists of different layers where the information ows only from one layer to the next layer. It has three layers: Input layer, Hidden Layer and Output Layer. Input layer has as much number of neurons as number of input parameters to the NN model, e.g. for the effort estimation task, inputs are size of the software. Hidden Layer, which gets inputs from input layer neurons, it is fully or partially connected to the input layer and fully connected with the Output Layer. Output Layer has one or more output neurons depending on the output of the model. For the effort estimation, output layer contains only single neuron which gives the result of the model in terms of effort (man months or man-hours) (Dave and Dutta, 2012). Each node in one layer is connected with a certain weight wij to every node in the following layer. Input nodes

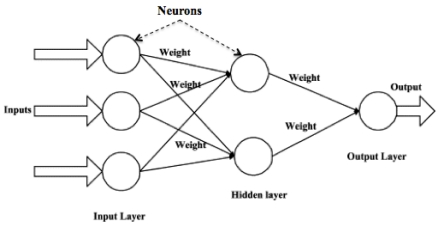

Fig. 1. Architecture of a Multilayer Perceptron (MLP)

distribute the signals to the nodes of the first hidden layer without processing it while nodes of hidden layers are neurons (or processing elements) with a nonlinear activation function (Trenn, 2008; Rosenblatt, 1962). Activation function is used to transform and squash the amplitude of the output signal of a neuron to some finite value. Some of the most commonly used activation functions are sigmoid, Tanh, and Gaussian (Karlik and Olgac, 2011). Weights are determined in the NN training process. This consists on initializing NN with random weights and gradually training the NN to capture the relationship between inputs and outputs by adjusting the weights based on the training dataset. Several training algorithms can be used namely the Resilient Back Propagation algorithm (RPROP).

2.3 Uncertainty in effort estimates

NN based effort estimation model suffers uncertainty from two sources: (1) The training data used in the elaboration of the NN that is typically noisy and incomplete. Project information collected by the companies can be evaluated subjectively and thus can be imprecise and noisy. Also, the dataset can not cover all possible software project input-output examples as it is just a sample from a large population of software projects. (2) The limitation of the model arising from the structure of the neural network that can be inappropriate, the training algorithm that may get stuck in a local rather than a global minimum of the error function, and that find solution that is valid only for regions sufficiently represented by the training data (Tiwari and Chatterjee, 2010; Papadopoulos et al., 2001).

To introduce uncertainty in NN based effort estimation, bootstrap procedure is employed which is based on resampling technique. The bootstrap is a computational procedure that uses intensive resampling with replacement, in order to reduce uncertainty (Efron and Tibshirani, 1994). Bootstrap technique has been used successfully with analogy based cost estimation (Angelis and Stamelos, 2000) and with semi-parametric software cost estimation model (LSEbA) (Mittas and Angelis, 2009). These works present methods of constructing Prediction Intervals (PIs) in order to describe the inherent uncertainty. PI consists of an upper and a lower effort value between which the future effort is expected to lie with a prescribed probability. Bootstrap technique has also been used in artificial neural network model development in many fields namely rainfall runoff modeling and hydrological modeling (Tiwari and Chatterjee, 2010).

To the best of our knowledge no studies have been reported in the software effort estimation literature that have used bootstrap based NN to take account of uncertainty and to make probabilistic estimates with PI.

3. BOOTSTRAPPING NEURAL NETWORK TO INTEGRATE UNCERTAINTY

The framework involved in our proposed method to take into account uncertainty present in dataset and NN effort estimation model, is shown in figure 2. This includes three processes: (1) Dataset preparation, (2) NN model structure determination and (3) NN bootstrapping.

3.1 Dataset preparation

The quality of a model depends on the quality of the dataset used in its elaboration. Thus, the processing of the dataset is an important task. This includes three steps: cleaning and features selection, transformation and division.

Dataset cleaning consists on discarding dataset related project attributes that are irrelevant for estimation, such as the project manager's name. Features or effort drivers are selected and determined using statistical tests such as Pearson correlation and one-way ANOVA (Laqrichi et al., 2013). Attributes that are rarely provided (for just 40% of projects) and projects with missing values in effort driver fields are discarded.

As NNs accept only numerical values (typically between 0 and 1), dataset transformation is required. Nominal values must be transformed into ordinal variables called dummies. Dummies are numeric variables that take the value of either 0 or 1. They represent the categories of an attribute (Garavaglia and Sharma, 1998). Moreover, numerical values must be transformed into values in the range of 0 to 1 to adapt them to the activation functions. For this purpose, normalization is used, as defined in (1), where a is one attribute value for a project, m and M are respectively the min and the max of this attribute values over the dataset projects.

$$a_{ij} \leftarrow \frac{a-m}{M-m} \tag{1}$$

Division consists in splitting randomly the dataset into two datasets one for training the NN and the others for testing it. The purpose is to test the performance of model on new data unused in the training stage. Generally, 80% of the dataset is allocated for model elaboration and 20% for model testing.

3.2 Neural Network model structure determination

This consists in designing the structure of the NN. It is one of the major task in elaborating a NN model. In this study, different design factors were considered in order to determine the appropriate NN structure. The defined NN structure parameters are:

- The number of inputs is the number of effort drivers.
- The number of outputs is the number of variables to estimate.
- The number of hidden layers is set to 1.
- The number of hidden nodes in each layer is generally between 1 and twice the number of inputs (Boetticher, 2001) which leads to a variety of NN structures.
- The activation functions considered are Identity in input and output layers and Sigmoid in hidden layer.

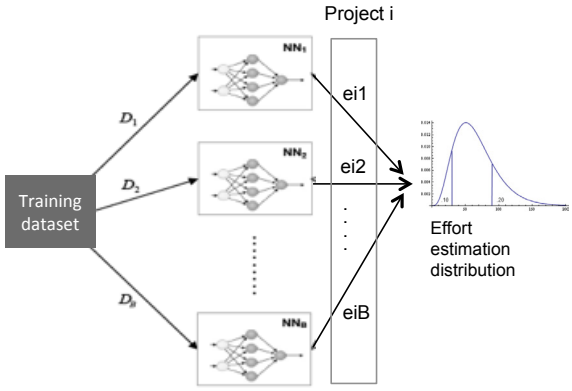

Fig. 3. Procedure to bootstrap NN

As there is no optimal and precise method to determine the most appropriate NN structure prior to training, the NN model structure is generally selected through a trial-and-error procedure (Tiwari and Chatterjee, 2010). That means each possible NN model structure is trained on the training dataset, tested on test dataset and finally the NN model structure giving the smallest error is selected. For this purpose, the K-fold cross validation technique can be used.

K-fold cross-Validation is a statistical method that can be used in evaluating and comparing different NN structures. The method of k-fold cross validation partitions the training set into k equally (or nearly equally) sized segments or folds. For each structure, subsequently k iterations of training and validation are performed, each time using one of the folds as the validation set and the remaining folds as the training set. The best NN model structure is then the one that has the smallest average error on the validation set (averaging over the k runs) (Refaeilzadeh et al., 2009).

3.3 Bootstrapping Neural Networks

In the case of regression problems, it is obvious to associate measures of confidence to the estimates. PIs are used for this purpose and can easily by evaluated using some defined formulas. However, for the NN estimation, there is no defined and precise manner to compute PIs (Mittas and Angelis, 2009). In this respect, we investigate the use of a simulation technique called bootstrap. This techniques is based on resampling with replacement of the available dataset. That's mean a project in the original dataset may be repeated many times in a generated sample. The purpose is to generate a large number of independent samples (Tiwari and Chatterjee, 2010).

The procedure to apply bootstrap in the context of NN effort estimation is shown in the figure 3 and the steps are given in the algorithm below.

(1) Generate a number B of "samples" from the original training dataset using resampling and replacement. B depends on the size of training dataset and ranges usually from 20 to 200 (Efron and Tibshirani, 1994).
(2) For each sample m, train a network with the same architecture. B trained networks are obtained.
(3) For each software project i in the test set, compute the set of estimates $e_i = \{e_{i1}, ...e_{im}, ...e_{iB}\}$ using the B trained network.

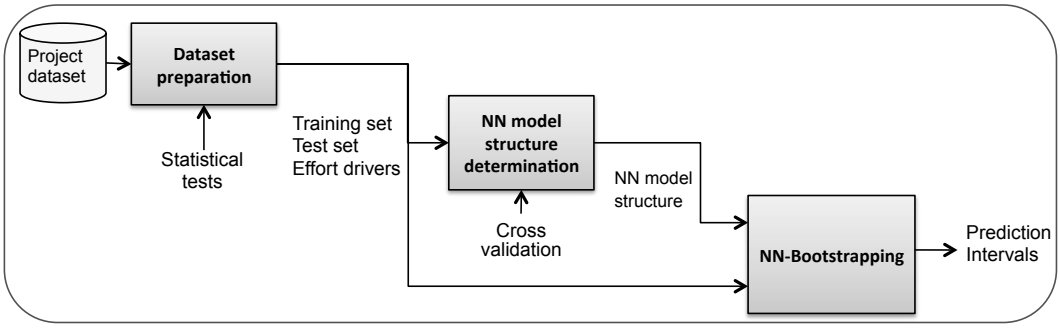

Fig. 2. Proposed method for integrating uncertainty in Neural Network effort estimation

The estimates produced by bootstrapped NN compose a distribution that can be used to compute the PI corresponding to a confidence level a. The PI is given by $[e_{a/2}, e_{1-a/2}]$, with $e_{a/2}$ and $e_{1-a/2}$ are efforts corresponding respectively to the $100(a/2)$ -th and the $100(1-a/2)$-th percentiles of the distribution. The most commonly used PI confidence levels in software estimation studies are 0.90 and 0.95 (Klas et al., 2011).

4. APPLICATION

The dataset used in our experimentation is from the International Software Benchmarking Standards Group (ISBSG R2011) considered as one of the largest and the more recent dataset for software effort estimation. It contains 5052 projects collected from companies worldwide from 1992 until 2009.

The analysis performed by (Huang and Han, 2008) on the ISBSG repository containing 1238 projects defines eight effort drivers that are: Adjusted Function Point (AFP), Max Team Size (MTS), Development Type (DT), Development Platform (DP), Language Type (LT), Used Methodology (UM), Methodology Acquired (MA), and Application Type (AT). The Person correlation test and the one-way ANOVA were adopted to test the relevance of these effort drivers based on the database of this study. As shown in table 1, two of effort drivers display important influence: AFP and MTS. For this experimentation, 252 projects were selected after discarding projects with information quality B and C, projects with missing values in effort drivers and atypical projects or outliers. Training and test dataset consist of respectively 202 and 50 projects.

Table 1. Results of Pearson correlation and one-way ANOVA on ISBSG

Feature	Correlation
AFP	0.287
MTS	0.264
DT	0.002
DP	0.002
LT	0.03
MA	0.01
AT	0.001

Statistical analysis like testing the normality of AFP, MTS and Effort showed that they are not normally distributed. These variables are transformed to logarithmic scale in order to respect some assumptions that are important in the elaboration of the model.

NN model structures can be denoted by (i:j:k), where i is the number of neurons in the input layer, j the number of neurons in the hidden layer, and k the number of neurons in the output layer. In this study, i and k are equal to 2 and 1 respectively and j ranges between 1 and 5. Each structure is trained on the training dataset, then it is tested on the test set and its Mean Magnitude of Relative Error (MMRE) is computed (see table 2). MMRE is the mean of estimation errors of projects of the test dataset. The structure with 4 nodes in hidden layer is found to be the best structure.

Table 2. MMRE for different NN model structures

NN structure	MMRE
2:1:1	0.46
2:2:1	0.39
2:3:1	0.36
2:4:1	0.35
2:5:1	0.37

The selected NN model structure is then bootstrapped using 100 samples generated from the training dataset. PIs corresponding to confidence level of 95% are then computed.

To evaluate the uncertainty estimates provided by PIs, some measures can be used namely the hit rate (HitR), the Relative PI width (rWidth) and the Actual effort Relative to PI (ARPI). The HitR provides the percentage proportion of estimates that fall within the PI, The rWidth expresses the precision of the PI and The ARPI inspects how the actual effort values are distributed relative to the PIs (Jorgensen et al., 2004).

The PIs provided by our method are compared with classical PIs based on linear regression effort estimation model (LR). From the measures presented in table 3, we observe that the LR gives slightly higher value of Hit rate than bootstrapped NN. Based only on this measure, it may be deduced that PIs estimates using LR is better than bootstrapped NN based PIs. However, the other measures should be examined as well before concluding on the performance of the two methods. In fact, the median rWidth of the LR is nearly two times larger than the bootstrapped NN. This mean that the LR provides very wider PIs which is difficult to use and interpret to make decision. Our proposed method achieves to provide narrowest PIs. In addition, the median ARPI measures show that the actual efforts is more close to the PIs midpoints in the our proposed method than in the LR

based method. This makes the bootstrap NN based PIs more reliable as the practitioners, in decision making, do not solely consider PIs but also PIs midpoints in a systematical and implicit way.

Taking account of these observations, our proposed method provides better results than the classical method based on LR. The bootstraped NN based PIs present well the uncertainty present in the effort estimation process and are realistic and reliable.

Table 3. Evaluating measures of effort PIs

	LR	Boostrapped NN
Hit rate (%)	72	68
Median rWidth	0.41	0.24
Median ARPI	0.13	0.09

5. CONCLUSION

Realistic effort estimation is an ongoing challenge for project managers, especially in an uncertain environment. This paper presents a method for a software effort estimation based on NN techniques. It allows uncertainty to be taken into account in effort estimates. NN based effort estimation model is used for its ability to model complex relationships, to be adapted to any project context and environment, and to be updated over time. Bootstrap simulation is used to generate independent samples in order to introduce uncertainty in NN based effort estimation. This method is probabilistic as it provides an effort estimation probability distribution from which a PI can be computed. Thus, it provides a better representation of reality, presents the uncertainty present in effort estimation process and offers a decision support tool for realistic effort estimation. The proposed method is applied on ISBSG R2011 dataset. The results show that the PIs provided by our method are realistic and reliable compared to LR based PIs. However, our method has some shortcoming as it uses the NN technique. First, NN model elaboration requires a good background in the subject. Furthermore, NN is a time consuming technique. Besides, it is considered as a black box which makes the justification and interpretation of estimates extremely difficult.

It is worth noting that there is another source of uncertainty that should be taken into account in the stage of utilization of the proposed bootstrapped NNs. It is related to the lack and imprecision of the available knowledge about the new software project being estimated. Monte Carlo simulation can be used to handle this uncertainty in future work (Mooney, 1997). For this purpose, the expert should assess the uncertainty of effort drivers through statistical distributions.

Another future step will be to evaluate and integrate software project risk in the proposed method in order to maximize its robustness. In fact, during the project development, risk events can occur, which may impact the progress of the project and lead to overruns. Hence, effort estimation PI should also consider the risk inherent in a software project, as well as uncertainty (Marmier et al., 2013). Future works also include the use of diverse large datasets to ensure the validity of the proposed method.

ACKNOWLEDGEMENTS

This work was funded by the Fonds Unique Interministriel (FUI) through the ProjEstimate project. We wish to acknowledge our gratitude and appreciation to all project partners for their contributions.

REFERENCES

Ahmed, M.A. and Muzaffar, Z. (2009). Handling imprecision and uncertainty in software development effort prediction: A type-2 fuzzy logic based framework. *Information and Software Technology*, 51(3), 640–654.

Angelis, L. and Stamelos, I. (2000). A simulation tool for efficient analogy based cost estimation. *Empirical software engineering*, 5(1), 35–68.

Basha, S. and Ponnurangam, D. (2010). Analysis of Empirical Software Effort Estimation Models. 7(3), 68–77.

Boehm, B. (2000). Software Cost Estimation with Cocomo II. *Prentice-Hall*.

Boetticher, G. (2001). An assessment of metric contribution in the construction of a neural network-based effort estimator. In *Second International Workshop on Soft Computing Applied to Software Engineering*.

Dave, V.S. and Dutta, K. (2012). Neural network based models for software effort estimation: a review. *Artificial Intelligence Review*, 42, 295–307.

Efron, B. and Tibshirani, R.J. (1994). *An Introduction to the Bootstrap*. CRC Press.

Garavaglia, S. and Sharma, A. (1998). A smart guide to dummy variables: four applications and a macro. In *Proceedings of the Northeast SAS Users Group Conference*. Citeseer.

Haykin, S.S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall International.

Huang, S.J. and Han, W.M. (2008). Exploring the relationship between software project duration and risk exposure: A cluster analysis. *Information & Management*, 45(3), 175–182.

Idri, A., Zakrani, A., and Zahi, A. (2010). Design of radial basis function neural networks for software effort estimation. *IJCSI International Journal of Computer Science Issues*, 7(4), 11–17.

Jorgensen, M. (2004). A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70(1), 37–60.

Jorgensen, M. and Sjoeberg, D.I.K. (2003). An effort prediction interval approach based on the empirical distribution of previous estimation accuracy. *Information and Software Technology*, 45(3), 123–136.

Jorgensen, M., Teigen, K.H., and Molkken, K. (2004). Better sure than safe? Over-confidence in judgement based software development effort prediction intervals. *Journal of Systems and Software*, 70(1-2), 79–93.

Karlik, B. and Olgac, A.V. (2011). Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4), 111–122.

Kitchenham, B. and Linkman, S. (1997). Estimates, uncertainty, and risk. *IEEE Software*, 14(3), 69–74.

Klas, M., Trendowicz, A., Ishigai, Y., and Nakao, H. (2011). Handling estimation uncertainty with boot-

strapping: Empirical evaluation in the context of hybrid prediction methods. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, 245–254.

Laqrichi, S., Marmier, F., and Gourc, D. (2013). Toward an effort estimation model for information system project integrating risk. *Proc. 22nd International Conference on Production Research (ICPR22)*.

Lee, A., Cheng, C.H., and Balakrishnan, J. (1998). Software development cost estimation: Integrating neural network with cluster analysis. *Information & Management*, 34(1), 1–9.

Marmier, F., Gourc, D., and Laarz, F. (2013). A risk oriented model to assess strategic decisions in new product development projects. *Decision Support Systems*, 56, 74–82.

Mittas, N. and Angelis, L. (2009). Bootstrap Confidence Intervals for Regression Error Characteristic Curves Evaluating the Prediction Error of Software Cost Estimation Models. In *AIAI Workshops*, 221–230.

Molokken, K. and Jorgensen, M. (2003). A review of software surveys on software effort estimation. In *International Symposium on Empirical Software Engineering (ISESE)*, 223–230. IEEE.

Mooney, C.Z. (1997). *Monte Carlo Simulation*. SAGE Publications.

Morgenshtern, O., Raz, T., and Dvir, D. (2007). Factors affecting duration and effort estimation errors in software development projects. *Information and Software Technology*, 49(8), 827–837.

Papadopoulos, G., Edwards, P., and Murray, A. (2001). Confidence estimation methods for neural networks: a practical comparison. *IEEE Transactions on Neural Networks*, 12(6), 1278–1287.

Park, H. and Baek, S. (2008). An empirical validation of a neural network model for software effort estimation. *Expert Systems with Applications*, 35(3), 929–937.

Refaeilzadeh, P., Tang, L., and Liu, H. (2009). Cross-Validation. In L. Liu and T. zsu (eds.), *Encyclopedia of Database Systems*, 532–538. Springer US.

Rosenblatt, F. (1962). *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Spartan Books.

Setiono, R., Dejaeger, K., Verbeke, W., Martens, D., and Baesens, B. (2010). Software Effort Prediction Using Regression Rule Extraction from Neural Networks. 45–52. IEEE.

Shepperd, M., Schofield, C., and Kitchenham, B. (1996). Effort estimation using analogy. In *Proceedings of the 18th international conference on Software engineering*, 170–178.

Srinivasan, K. and Fisher, D. (1995). Machine learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering*, 21(2), 126–137.

Tiwari, M.K. and Chatterjee, C. (2010). Uncertainty assessment and ensemble flood forecasting using bootstrap based artificial neural networks (BANNs). *Journal of Hydrology*, 382(14), 20–33.

Trenn, S. (2008). Multilayer perceptrons: approximation order and necessary number of hidden units. *IEEE transactions on neural networks*, 19(5), 836–844.