Estimating Software Effort Using Neural Network: An Experimental Investigation

P. Suresh Kumar and H. S. Behera

Abstract Software companies develop various softwares at the same time. It is very critical task that is to be managed by project managers. Completion of a project is purely dependent on various parameters such as time, cost and staff. By these parameters, project planning will be done by the project managers. Software effort estimation is a fundamental and emerging aspect for software companies in developing a software. If we estimate the effort properly, it will control the project cost as well as time. This paper is presented by comparing various models such as KNN, SVM, NN, RF and back propagation algorithm using feed forward neural network by using Orange data mining tool. The proposed models are evaluated using COCOMO'81 dataset having 63 projects and Desharnais dataset having 81 projects. Estimation results are evident that the back propagation-based approach is a suitable model as compared to the remaining considered approach.

Keywords Software effort estimation · Back propagation algorithm · K-nearest neighbor regression · Support vector machine · Neural network · Random forest

1 Introduction

In software development process, software effort estimation (SEE) plays a crucial and very important role. Calculating correct SEE is a difficult task, but it is essential in software development [1]. If estimation is not done properly, then this effects in software project failure. Main complication lies in anticipating parameters such as exact estimation of effort, duration and cost. Still several software companies are using expert judgment for predicting SEE which leads untrue estimations and serious overruns in the time schedule of their projects [2]. Nowadays, regulating software expenses in the development of software is an important concern in competitive world. In the early days, software project managers must be able to estimate the effort and cost [3]. To estimate effort, project managers follow dependable techniques for

P. Suresh Kumar (✉) · H. S. Behera
Department of IT, Veer Surendra Sai University of Technology, Burla 768018, India

A. K. Das et al. (eds.), *Computational Intelligence in Pattern Recognition*,
Advances in Intelligent Systems and Computing 1120,
https://doi.org/10.1007/978-981-15-2449-3_14

operating activities such as resource allocation, feasibility studies, project bidding, planning and cost estimation [4].

There are many estimation models available to calculate the effort such as expert judgment, algorithmic and non-algorithmic. Expert judgment, in this the estimation depends upon the similar projects which are completed. If the completed projects' parameters are almost same as the current project, then it will be easy to calculate the effort of the current project [5]. Algorithmic models are parametric in this effort estimation and are predicted using some fixed formed formulas that are parameterized from historic data. Some of the algorithmic approaches are COCOMO [2], Putnam's SLIM [6], Albrecht's Function Points [7] among others. Due to the limitations in algorithmic models, non-algorithmic models came into the scenario that are based on soft computing and machine learning models. In spite of those models, new models are being proposed day by day to find accurate estimation [8].

Nowadays for predicting software effort estimation, many authors are exploring toward soft computing techniques. Since 1960s, soft computing is extensively known and applied. The main goal of soft computing is to manage tolerance from incomplete data and to make a decision from imprecision, uncertain data. When there is no solution for mathematical models, soft computing alone will solve such type of problems. In calculating software effort estimation, neural networks produce good estimations among all soft computing techniques [9].

There are various models available in the artificial neural network, such as multilayer perceptron, general regression neural network, radial basis function neural network and cascade correlation neural network. In these, there is no clarity on which model is giving the best results. To train the neural network, there are many algorithms developed in recent years. From the literature, it is evident to note that back propagation using gradient decent learning is a suitable method to train the network [10].

Rest of the paper is categorized into six sections. In Sect. 2, the literature review of various artificial neural network models is explained. Section 3 describes about basic preliminary, which explains artificial neural architecture. Section 4 explains about flow chart of the proposed work. In Sect. 5, the experimental setup which explains system models dataset description and all is given. In Sect. 6, the results are shown . In Sect. 7, the conclusion of work and some further developments are given.

2 Literature Review

deBarcelos Tronto et al. [11] investigated the prediction of artificial neural network and regression-based models. The main goal of the author is to simplify models to achieve good effort estimation. They used COCOMO and ISBSG datasets. In their research, they have considered the main measures for accuracy as MMRE and R^2. From the results, they compared artificial neural network to linear regression and concluded that the neural network performed better as compared to linear regression.

Idri et al. [12] used two clustering techniques such as in designing radial basis function neural network for estimation such as C-means and APC-III: They used COCOMO'81 dataset for training as well as testing. From the accuracy measures such as MMRE and PRED, it is RBFN using C-means and APC-III are efficient compared to others.

El-Sebakhy [13] proposed a new intelligent model called the functional network to estimate the effort. The author compared the performance results with previous models such as basic multilayer perceptron and nonlinear statistical regression for software effort estimation. From the results, it is proved that the performance in functional point gave good results as compared to the remaining methods. In the study, they used Kemerer, IBM, Hallmark datasets and evaluated MAPE values.

For better performance, Li et al. [14] added learning ability to a nonlinear adjustment mechanism which is more flexible and included categorical features. Author implemented nonlinear adjusted analogy-based estimation method by adding a learning ability using artificial neural networks. This proposed NABE using ANN is validated using four real-world datasets and compared the results with three other models called ANN, CART and SWR. Author concluded that the proposed method performs well in terms of MMRE and PRED (0.25).

Ghose et al. [15] stated that homogeneous dataset results are having better accuracy when compared to irrelevant and chaotic methods in software effort estimation. They explored the accuracy of different neural networks from the dataset developed by Lopez et al. [16] They adopted mostly used evolution factors such as MMRE, MdMRE, BRE and Pred (0.25). From the work, they concluded the generalized regression neural network model gave good results among other neural network models.

Kaushik et al. [17] designed feed forward-based back propagation neural network model for estimating the effort. In the designed neural network, they used identity function at neural network, and sigmoid function is applied at hidden as well as output layer. It is designed in such a manner that it can accommodate the neural network with COCOMO model. This proposed model enhanced the software effort estimation prediction. Proposed model is tested using COCOMO and COCOMO NASA 2 datasets. Results are compared with COCOMO model, and stated combination of the traditional COCOMO model with neural network approach gives good results.

Benala et al. [18] established hybrid method which uses unsupervised learning with functional artificial neural network (UKW/DBSCAN) for predicting software effort estimation. In FLANN, they used robust computational model Chebyshev polynomial for functional expansion to check the performance. They used COCOMO 81, NASA and Desharnais for empirical evaluation. From their experimentation, it is concluded that the proposed model produces good results as compared to conventional FLANN.

Faruk and Nevcihan [19] combined software effort estimation methods like COCOMO and ANN with K-means, with the calculation of possible boundaries. They calculated MRE and MMRE and evaluated possible boundaries. In experimental evaluation, they used COCOMO and NASA datasets. These boundaries help

managers in decision making. Experimental results are in support to the proposed method in terms of good results compared to COCOMO and ANN.

Satapathy et al. [20] considered total story points and project velocity to forecast the development effort. Results were optimized with various neural networks such as RENN, PNN, GMDHN and CCNN. They used project data for experimentation. These are validated empirically to know the performance of these models, and it is seen that cascade neural network performed well among these models.

Lopez-Martin [21] compared various models such as RBFNN with regression, statistical, feed forward MLP and GRNN. With the help of ISBSG dataset, they concluded that the results of RBFNN are quite encouraging.

Nassif and Azzeh [22] proposed hybrid model where two different models such as SVM and RBNN are used for classification and prediction, respectively. They proposed the model from the observation of industrial and student projects and compared the above proposed model with UCP prediction model. The results demonstrated that the proposed model is efficient over all datasets.

Rijwani and Jain [23] used base model of artificial neural network using multilayer feed forward neural network and is being trained by using back propagation neural network. They used COCOMO dataset, and empirical evaluation was done by using MSE and MMRE.

3 Basic Preliminary—Artificial Neural Network

Artificial neural network is a technique which is an efficient information processing system and acts like human brain. We can say, ANN is a mathematical model of a human brain. From the behavior, ANN can be categorized by their ability to learn, recall and training pattern [24]. Artificial neurons are interconnected to one another by a connected link: These connected links are having their related weights with input signal's information. Neural net will use the information and solve particular problem. Each neuron will have an activation level, which acts as activation function. There are many activation functions available such as Gaussian, linear, tanh and sigmoid. In these, sigmoid is the most frequently used activation function for neural networks [17].

By architecture, ANN is classified into two categories: feed forward and feedback. Feed forward can have two layers called input layer and output layer, where each layer contains one or more neurons. There is an optional layer named hidden layer which is in between input and output layer. In feed forward neural network, signal flow is from input layer to output layer through hidden layers. From the connection of neurons, it groups the behavior of neurons and decides the output. In feedback neural network, to get the best results internally, the output passes back to the network. This feeds information to input or other hidden layers [25].

Performance of artificial neural network is mainly based on the learning algorithm and activation function used. Normally in the case of software effort estimation, endorsed architecture, learning algorithm and activation functions are feed forward multilayer perceptron, back propagation algorithm and sigmoid function, respectively. Most of the people are using back propagation algorithm which is using

gradient descent in supervised learning. In respective weights, this algorithm is calculating the gradient. This gradient is provided to some optimization method so that the weights are updated [10].

Artificial neural network works well in the following problems [26]:

- It will establish some association among patterns
- It can handle large dataset and it is diverse
- It can handle when the variable numbers are high
- It can understand the relationships among variables

Sample architecture of artificial neural network is shown in Fig. 1. Usually, ANN has three layers as shown in figure such as input, hidden (may have one or more) and output layers. We can call this as multilayer perceptron as it is having multiple numbers of layers. Number of neurons in the input layer can be taken according to the input parameters. If we consider software effort estimation, input parameters are FP, LOC, effort drivers like complexity, database size, language experience and programmer's capability. In hidden layer, it will take the input as some important patterns from input layer. It will make the network fast and efficient by considering only important information. It is fully or partially connected to the input layer, and it is fully connected to the output layer. For effort estimation, output layer has only one neuron that is person-hour or person-months.

As shown in Fig. 1, let us consider input layer is having 'n' neurons ($x1$, $x2$ …. xn), hidden layer is having 'n' neurons ($y1$, $y1$ …. yn), and output layer is having only one neuron (z). Above architecture resembles mathematical representation of human brain. In the above architecture net input at hidden neuron is calculated as shown in Eq. 1.

$$y_{in} = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n = \sum_{i=1}^{n} x_i w_i \tag{1}$$

where

'i' represents ith processing element.

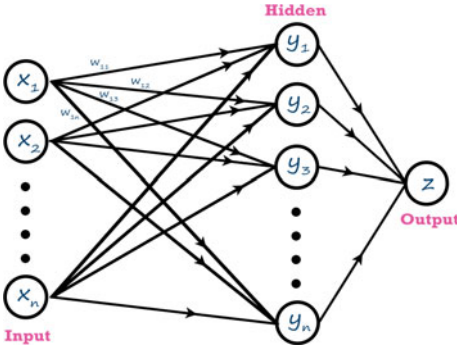

Fig. 1 Architecture of ANN

Some activation function is applied on y_{in} to get 'y,' and this is supplied as input to the neuron in the next layer which is shown in Eq. 2.

$$y = f(y_{in}) \tag{2}$$

4 Work Done

In the proposed work, we utilized different techniques identified with machine learning such as KNN, SVM, neural network, random forest and back propagation algorithm to predict software effort estimation. In this work, we have gone through various phases including

- Data collection: datasets are collected from PROMISE repository,
- Preprocessing: where we converted the raw data into useful clean dataset,
- Classification models with identification parameters,
- Performance evaluation: using various machine learning algorithms evaluate performance measures,
- The sequence of phases starting from data collection through performance evaluation constitutes an iteration. Iterations are repeated till the evolutionary factors are satisfied after adjusting the individual parameters corresponding to each model.

The proposed system is presented in Fig. 2 below.

5 Experimental Setup

This section explained about Orange data mining tool, dataset description with different parameter changing in various techniques and their predicted values and the experiment description in back propagation neural network and methodology.

5.1 Orange—Visual Programming

Experimentation has been done using the 'Orange' tool [27], which is an open-source software used in data mining, data visualization and machine learning. It performs visual programming based on user-defined or predefined components called widgets. It is implemented through an interface, where workflows are created by connecting predefined or user-defined components using communicating channels. Predefined components include number of graphical widgets having a specific functionality that might include reading data, selecting data features, showing data table, training predictors and cross-validating them. Being a comprehensive and component-based

Fig. 2 Proposed framework

framework used in machine learning and data mining, the Orange tool is the preferred tool by both researchers and academicians [28].

5.2 Dataset Description

COCOMO'81 dataset is publicly available in PROMISE repository analyzed by Boehm [29]. In the present work, we used COCOMO'81 having 63 tuples. Each tuple represents the metrics associated with calculation of software effort estimation for each individual project. Out of the available 17 numeric attributes, 15 are effort multipliers and the remaining two being lines of code and actual development effort. These 15 include analysts capability, programmers capability, application experience, modern programming practices, use of software tools, virtual machine experience, language experience, schedule constraint, main memory constraint, data base size, time constraint for CPU, turnaround time, machine volatility, process complexity and required software reliability. There is no feature missing value in the dataset.

Desharnais dataset is also available in PROMISE repository, and it is donated by Desharnais [30]. This is having 81 tuples and four of these contain missing values,

and these are excluded in this experiment. Each tuple is a project consisting of 12 attributes, including Project, TeamExp, ManagerExp, YearEnd, Length, Effort, Transactions, Entities, PointsAdjust, Envergure, PointsNonAjust and Language.

5.3 Configuring the Various Models for Predicting the Software Effort Estimate Using Orange Tool

This section showed the investigated results on various techniques such as KNN, SVM, NN and RF using Orange tool by providing various parameters on COCOMO'81 and Desharnais datasets. From investigation, better predicted effort is considered.

5.3.1 K-Nearest Neighbor Regression (KNN) Method

KNN is proposed in 1978 by Devroye [31]. KNN is used in both classification as well as regression; we are using this method which is originally in artificial intelligence. In cross-validation, it will select appropriate k-value. KNN is a linear neural network searching algorithm. KNN classifies the objects from near training example from the feature space. Here, classification of object is done by using majority votes of nearest neighbors. By using feature similarity, we can predict new data points. In the proposed work, we experimented with different parameters (Table 1), and final parameter initializations are closest neighbor: 3, metric: Euclidean, weight: uniform.

5.3.2 Support Vector Machine (SVM) Method

Support vector machine presented by Vapnik [32] is based on kernel-based learning algorithm. SVM is used in numerous applications for classification and regression. By using SVM, we can reduce the generalization error with structural risk minimization. SVM concurrently minimizes the empirical error while maximizing the geometric margin [33]. In this paper, we reviewed various parameters which are shown in Table 2. We considered v-SVM type, polynomial kernel and set the numerical tolerance to 0.0010 and iteration limit to 50.

5.3.3 Neural Network (NN) Method

Neural networks are one of the best architectures used in software effort estimation. We build our model with multilayer perceptron, which is a class of feed forward neural network. MLP is acceptable for classification and prediction problems where inputs are assigned to a class. These are used to learn mapping from input layer to

Table 1 Effort estimation computed with different parameters in KNN using COCOMO'81 and Desharnais dataset

Project Id	No of neighbor	Metric	Computed effort using COCOMO'81	Effort from COCOMO'81	Computed effort using Desharnais	Effort from Desharnais
1	3	Euclidean	1227	2040.0	4786	5152
		Manhattan	773.7		3927	
		Chebyshev	1227		5187	
		Mahalanobis	1246		7145	
	4	Euclidean	980.2		4546	
		Manhattan	610.8		4354	
		Chebyshev	931		4797	
		Mahalanobis	944.8		5560	
	5	Euclidean	808.6		4633	
		Manhattan	808.6		4764	
		Chebyshev	755.8		5194	
		Mahalanobis	833.2		6275	
8	3	Euclidean	1032	1075.0	5313	3913
		Manhattan	818.7		3927	
		Chebyshev	914.3		2977	
		Mahalanobis	818.7		2380	
	4	Euclidean	925.2		4196	
		Manhattan	629		4522	
		Chebyshev	718.2		4196	
		Mahalanobis	617.5		4048	
	5	Euclidean	824.8		4171	
		Manhattan	522.8		4431	
		Chebyshev	592.2		3787	
		Mahalanobis	706.6		4014	

output layer. It is implemented through the procedure, which will do many times to set weights that will minimize the mean square error [34]. In this work, we examined with various parameters, shown in Table 3. We assigned seven hidden layers, tanh activation function, learning rate 0.0005 and maximal number of iterations 50.

5.3.4 Random Forest (RF) Method

Random forest is a technique which is used for classification and regression basis. In training period, it generates a number of decision trees which are later used to select the final class based on the mode of class. To obtain satisfactory results, rather than

Table 2 Effort estimation computed with different parameters in SVM using COCOMO'81 and Desharnais dataset

Project Id	SVM type	Kernal	Computed effort using COCOMO'81	Effort from COCOMO'81	Computed effort using Desharnais	Effort from Desharnais
1	SVM	Linear	177.5	2040.0	3738	5152
		Polynomial	101.3		3655	
		RBF	98.6		3600	
		Sigmoid	98.2		3667	
	v-SVM	Linear	321.7		4207	
		Polynomial	254.4		4177	
		RBF	248.3		4137	
		Sigmoid	252.6		4133	
8	SVM	Linear	187.5	1075.0	3718	3913
		Polynomial	118.7		3654	
		RBF	98.2		3657	
		Sigmoid	98.6		3661	
	v-SVM	Linear	341.5		4163	
		Polynomial	277		4175	
		RBF	248		4136	
		Sigmoid	252.4		4129	

combining the results from independent decision tree models, results of models of the same type are combined [35]. Table 4 shows the parameters considered for random forest approach. We considered six number of trees and controlled the growth by stop splitting the subsets which are smaller than 2.

5.4 Details of Software Effort Estimation Using BPNN

Back propagation neural network is implemented using Python3.x and NumPy package. For implementation, we have used COCOMO'81 and Desharnais datasets having 63 and 81 records, respectively. From these, 80% records are allocated for training and 20% records for training. All are numeric and no missing values. In this implementation, a simple neural network with 15 input nodes (each node for each feature), seven nodes in hidden layer (determined by trial and error) and two output nodes for COCOMO'81 and 11 input nodes, 7 nodes in hidden layer and 1 output node in output layer has been developed.

This back propagation algorithm is iterative and set max epochs to 50, activation functions as sigmoid and learning rate to 0.05 that will control each weight and change

Table 3 Effort estimation computed with different parameters in NN using COCOMO'81 and Desharnais dataset

Project Id	Optimizer	Activity function	Computed effort using COCOMO'81	Effort from COCOMO'81	Computed effort using Desharnais	Effort from Desharnais
1	L-BFGS-B	Identity	2310.9	2040.0	6458	5152
		Logistic	2167.3		6221	
		tanh	1617.4		9342	
		ReLu	2075.9		6785	
	SGD	Identity	nan		nan	
		Logistic	1141.7		5478	
		tanh	1378.4		7025	
		ReLu	nan		nan	
	Adam	Identity	−0.5		2	
		Logistic	0.6		1	
		tanh	−0.2		1	
		ReLu	1.6		2	
8	L-BFGS-B	Identity	2310.9	1075.0	4283	3913
		Logistic	1310.3		6221	
		tanh	831		4512	
		ReLu	1234.1		3774	
	SGD	Identity	nan		nan	
		Logistic	1099.5		5478	
		tanh	1378.4		4031	
		ReLu	1378.4		nan	
	Adam	Identity	−0.5		1	
		Logistic	0.6		1	
		tanh	−0.2		1	
		ReLu	1.6		2	

in bias value. Small learning rate will effect slow in training but gives significant training, and large learning rate effects quicker training, but it may exceed good weight and bias value. Along with learning rate, magnitude of the gradient is used that will declare whether to increase or decrease the bias and weight as well.

Table 4 Effort estimation computed with different parameters in RF using COCOMO'81 and Desharnais dataset

Project Id	No of trees	Split subsets	Computed effort using COCOMO'81	Effort from COCOMO'81	Computed effort using Desharnais	Effort from Desharnais
1	5	<2	1952	2040.0	5152	5152
		<3	1792.4		5272	
		<4	1305.8		5064	
	6	<2	3276.3		5152	
		<3	1863.9		5183	
		<4	1880.6		5010	
	7	<2	3099.7		5127	
		<3	3300		5154	
		<4	2447.8		5005	
8	5	<2	1075	1075.0	3913	3913
		<3	977		3882	
		<4	1025.7		3886	
	6	<2	716.7		3913	
		<3	783.1		4025	
		<4	788.8		4028	
	7	<2	927.1		3913	
		<3	1022.5		4009	
		<4	882.9		4016	

5.5 Methodology

By using Orange tool, we classified COCOMO'81, Desharnais datasets and predicted the effort with models being KNN, SVM, neural networks and random forest. The developing environment used to propose the method such as KNN, SVM, NN and RF is using Orange tool and back propagation model using Python. While preparing the model, the datasets are segregated as 80% and 20%. Sigmoid function is used as activation function in the network. Best fit neural network architecture is 15-7-1. Set learning rate is fixed as 0.05. MMRE is used to conclude the correctness of prediction in training, testing as well as validation.

6 Results

Tables 5 and 6 show the variations between effort obtained from COCOMO'81 and Desharnais datasets, respectively, and effort computed using various techniques

Table 5 Effort estimation computed different models using COCOMO dataset

Project id	Effort obtained by COCOMO dataset	Computed effort with different techniques				
		KNN	SVM	NN	RF	BPNN
1	2040.0	1227.0	254.4	1617.4	1738.3	2019.61
8	1075.0	1032.0	277.0	831.0	733.7	1064.26
11	218.0	182.7	247.1	831.0	216.2	215.83
18	11400.0	4152.0	249.2	1617.4	7386.3	11286.01
23	539.0	497.7	247.3	831.0	422.0	533.62
25	523.0	1059.7	248.1	1617.4	751.2	517.78
31	1063.0	559.0	285.5	831.0	1788.8	1052.38
34	230.0	279.7	246.4	1617.4	192.9	227.71
45	106.0	92.3	245.9	1216.3	92.3	104.95
49	156.0	186.0	246.0	129.4	164.0	154.45

Table 6 Effort estimation computed different models using Desharnais dataset

Project Id	Effort obtained by Desharnais dataset	Computed effort with different techniques				
		KNN	SVM	NN	RF	BPNN
1	5152	4768	4177	4630	4899	5100.49
8	3913	5313	4175	4630	4459	3873.88
11	4067	6991	4177	5641	6206	4026.34
18	1617	5388	4180	4630	3824	3402.64
23	5775	7653	4178	5641	5752	5717.26
25	3983	4800	4177	3846	3846	3943.18
31	3927	4188	4175	4630	4401	3887.74
34	6405	4153	4176	5641	6231	6340.96
45	6699	4718	4177	5641	7628	6632.02
49	2331	1584	4169	4630	4734	2307.7

such as K-nearest neighbor regression, support vector machine, neural networks, random forest and back propagation algorithm with above-said parameters. It is clearly noticed that prediction using back propagation is relatively good compared to other techniques.

7 Conclusion

The major models used for software effort estimation include KNN, SVM, NN, RF and BPNN. The aim of this paper is to compare these models and determine which among them is most suitable for software effort estimation for the considered datasets

Desharnais and COCOMO'81. The Orange tool is used for comparing the prediction performance of KNN, SVM, NN and RF. Among the models experimented using the Orange tool, it is found that software effort prediction using KNN is slightly better when compared with other models.

A Python implementation of back propagation algorithm applied on feed forward neural network is done, where we used 80% projects for training and 20% projects for testing.

An overall comparison of these models indicates that the software effort estimate obtained using BPNN is significantly better when compared with all the other models.

References

1. Murillo-Morera, J., Quesada-López, C., Castro-Herrera, C., et al.: A genetic algorithm based framework for software effort prediction. J. Softw. Eng. Res. Dev. **5**(1), Dec (2017). https://doi.org/10.1186/s40411-017-0037-x
2. Boehm, Barry W.: Software engineering economics. Prentice-Hall, Englewood Cliffs, N.J. (1981)
3. Huang, Jianglin, Li, Yan-Fu, Xie, Min: An empirical analysis of data preprocessing for machine learning-based software cost estimation. Inf. Softw. Technol. **67**, 108–127 (2015). https://doi.org/10.1016/j.infsof.2015.07.004
4. Benala, T.R., Mall, R.: DABE: Differential evolution in analogy-based software development effort estimation. Swarm Evol. Comput. 38, 158–172, ISSN 2210-6502 (2018). https://doi.org/10.1016/j.swevo.2017.07.009
5. Thamarai, I., Murugavalli, S.: Using differential evolution in the prediction of software effort. In: Fourth International Conference on Advanced Computing (ICoAC), pp. 1–3, Chennai (2012). https://doi.org/10.1109/icoac.2012.6416816
6. Putnam, L.H.: A general empirical solution to the macro software sizing and estimating problem. IEEE Trans. Softw. Eng. **SE-4**(4), 345–361, July (1978). https://doi.org/10.1109/tse.1978.23152
7. Albrecht, A.J., Gaffney, J.E.: Software function, source lines of code, and development effort prediction: a software science validation. IEEE Trans. Softw. Eng. **SE-9**(6), 639–648, Nov (1983). https://doi.org/10.1109/tse.1983.235271
8. Ferrucci, F., Gravino, C., Oliveto, R., Sarro, F., Genetic programming for effort estimation: an analysis of the impact of different fitness functions. In: IEEE Second International Symposium on Search Based Software Engineering (SSBSE), pp. 89–98, (2010). https://doi.org/10.1109/ssbse.2010.20
9. Suresh Kumar, P., Behera, H.S.: Role of soft computing techniques in software effort estimation: an analytical study. In: Das A., Nayak J., Naik B., Pati S., Pelusi D. (eds.) Computational intelligence in pattern recognition: Advances in Intelligent Systems and Computing vol 999, pp. 807–831. https://doi.org/10.1007/978-981-13-9042-5_70
10. Nassif, A.B. et al.: Neural network models for software development effort estimation: a comparative study. Neural Comput. Appl. **27**(8), 2369–2381 (2016). https://doi.org/10.1007/s00521-015-2127-1
11. deBarcelos Tronto, I.F., da Silva, J.D.S., Sant'Anna, N.: Comparison of artificial neural network and regression models in Software Effort Estimation. In: IEEE International Joint Conference on Neural Networks (IJCNN), pp. 771–776 (2007). https://doi.org/10.1109/ijcnn.2007.4371055
12. Idri, A., Abran, A., Mbarki, S.: An experiment on the design of radial basis function neural networks for software cost estimation. IEEE Int. Conf. Inf. Commun. Technol. **1**, 1612–1617 (2006). https://doi.org/10.1109/ICTTA.2006.1684625

13. El-Sebakhy, E.A.: New computational intelligence paradigm for estimating the software project effort. In: IEEE 22nd International Conference on Advanced Information Networking and Applications Workshops (AINAW 2008), pp. 621–627 (2008). https://doi.org/10.1109/waina.2008.257
14. Li, Y.F., Xie, M., Goh, T.N.: A study of the non-linear adjustment for analogy based software cost estimation. Empir. Softw. Eng. **14**(6), 603–643 (2009). https://doi.org/10.1007/s10664-008-9104-6
15. Ghose, M.K., Bhatnagar, R., Bhattacharjee, V.: Comparing some neural network models for software development effort prediction. In: IEEE 2nd National Conference on Emerging Trends and Applications in Computer Science (NCETACS), pp. 1–4 (2011). https://doi.org/10.1109/ncetacs.2011.5751391
16. Lopez-Martin, C., Yanez-Marquez, C., Gutierrez-Tornes, A.: Predictive accuracy comparison of fuzzy models for software development effort of small programs. J. Syst. Softw. **81**(6), 949–960 (2008). https://doi.org/10.1016/j.jss.2007.08.027
17. Kaushik, A., Soni, A.K., Soni, R.: An adaptive learning approach to software cost estimation. In: National IEEE Conference on Computing and Communication Systems (NCCCS), pp. 1–6 (2012). https://doi.org/10.1109/ncccs.2012.6413029
18. Benala, T.R., Dehuri, S., Mall, R., ChinnaBabu, K.: Software effort prediction using unsupervised learning (clustering) and functional link artificial neural networks. In: IEEE World Congress on Information and Communication Technologies (WICT), pp. 115–120 (2012). https://doi.org/10.1007/978-3-642-32341-6_1
19. Saraç, Ö.F., Duru, N.: A novel method for software effort estimation: Estimating with boundaries. In: IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA), pp. 1–5 (2013). https://doi.org/10.1109/inista.2013.6577643
20. Aditi, P., Satapathy, S.M., Rath, S.K.: Empirical validation of neural network models for agile software effort estimation based on story points. Procedia Comput. Sci. **57**, 772–781, ISSN 1877-0509 (2015). https://doi.org/10.1016/j.procs.2015.07.474
21. López-Martín, C.: Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects. Appl. Soft Comput. **27**, 434–449, ISSN 1568-4946 (2015). https://doi.org/10.1016/j.asoc.2014.10.033
22. Azzeh, M., Nassif, A.B.: A hybrid model for estimating software project effort from Use Case Points. Appl. Soft Comput. **49**, 981–989, ISSN 1568-4946 (2016). https://doi.org/10.1016/j.asoc.2016.05.008
23. Poonam Rijwani, S.J: Enhanced software effort estimation using multi layered feed. In: Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016), Jaipur: Elsevier, pp. 307–312 (2016). https://doi.org/10.1016/j.procs.2016.06.073
24. Ajitha, S., Kumar, T.S., Geetha, D.E., Kanth, K.R.: Neural network model for software size estimation using a use case point approach. In: IEEE International Conference on Industrial and Information Systems (ICIIS), pp. 372–376 (2010). https://doi.org/10.1109/iciinfs.2010.5578675
25. Kalichanin-Balich, I., Lopez-Martin, C.: Applying a feedforward neural network for predicting software development effort of short-scale projects. In: IEEE Eighth ACIS International Conference on Software Engineering Research, Management and Applications (SERA), pp. 269–275 (2010). https://doi.org/10.1109/sera.2010.41
26. Dave, V., Dutta, K.: Neural network based models for software effort estimation: A review. Artif. Intell. Rev. **42**, 295–307, Aug (2014). https://doi.org/10.1007/s10462-012-9339-x
27. Demšar, J., Zupan, B.: Orange: data mining fruitful and fun—a historical perspective. Inform. **37**, 55–60 (2013)
28. Idri, A., Mbarki, S., Abran, A.: Validating and understanding software cost estimation models based on neural networks. In: IEEE International Conference on Information and Communication Technologies: From Theory to Applications, pp. 433–434 (2004). https://doi.org/10.1109/ictta.2004.1307817
29. Boehm, B.W.: Software Engineering Economics. IEEE Trans. Softw. Eng. **SE-10**, 4–21 (1981)

30. Desharnais, J.M.: Analyse statistique de la productivitie des projets informatique a partie de latechnique des point des foncti\ on. University of Montreal, Masters Thesis (1989)
31. Devroye, L.: The uniform convergence of nearest neighbor regression function estimators and their application in optimization. IEEE Trans. Inf. Theory **24**(2), 142–151, March (1978). https://doi.org/10.1109/TIT.1978.1055865
32. Vapnik, V.N.: The nature of statistical learning theory. In: Springer-Verlag, ISBN: 0-387-94559-8 (1995)
33. Braga, P.L., Oliveira, A.L.I., Meira, S.R.: A GA-based feature selection and parameters optimization for support vector regression applied to software effort estimation. In: Proceedings of the 2008 ACM symposium on Applied computing (SAC '08), pp. 1788–1792 (2008). https://doi.org/10.1145/1363686.1364116
34. Sharma, N., Litoriya, R.: Incorporating data mining techniques on software cost estimation: Validation and improvement. Procedia Technol. **1,** 65–71 (2012). https://doi.org/10.1016/j.protcy.2012.02.013
35. Satapathy, S.M., Acharya, B.P., Rath, S.K.: Early stage software effort estimation using random forest technique based on use case points. IET Softw. **10**(1), 10–17 (2016). https://doi.org/10.1049/iet-sen.2014.0122