Estimating Software Effort and Function Point Using Regression, Support Vector Machine and Artificial Neural Networks Models

Sultan Aljahdali
Computer Science Department
Taif University
Taif, Saudi Arabia
aljahdali@tu.edu.sa

Alaa F. Sheta
Computers and Systems Department
Electronics Research Institute
Giza, Egypt
asheta66@gmail.com

Narayan C. Debnath
Computer Science Department
Winona State University
Winona, MN 55987, USA
ndebnath@winona.edu

***Abstract*—Accurate computation of software effort, cost and time required ahead would greatly reduce risk and maximize profit. Estimating software effort or computing the required function point helps project manager to better estimate the time and budget required for a project. Many statistical models were proposed in the past. These models suffer many problems related to parameter estimation and structure determination of the models. In this paper we presents two models for software effort estimation and one model for function points using Linear Regression (LR), Support Vector Machines (SVM) and Artificial Neural Networks (ANN). The proposed models have number of inputs and single output. The first model utilizes the Source Line Of Code (KLOC) as inputs; while the second model utilize the KLOC and development Methodology (ME) as inputs to estimate the Effort (E); while the third model utilize the Inputs, Outputs, Files, and User Inquiries as inputs to estimate the Function Point (FP). The proposed SVM and ANN models show better estimation capabilities compared to linear regression model models. These models are capable of providing better assistant to software project manager in computing the effort required of the number of function points.**

I. Introduction

Computing the estimate of a Software system is a common problem for software engineers. It is essential for any software development process to accurately compute the project budget, project time and develop a plane for implementation [1]–[4]. These are serious practices in the software industry, since poor budgeting and planning often has serious results. Some applications include Military Application, NASA Space Shuttle systems, Air Force and business for huge Enterprises. It was mentioned that NASA and Air Force projects spent about 50% of their development cost in software development.

In the past few decades, it was noticed that software community experiences many challenges associated to computing the exact software resource prediction. Many models were proposed to handle this problem. These models can be classified as theoretical such as the Putnam's model [5], [6] and empirical models such as the Walston and Fleix [7]. Theoretical models can be characterized by a formulas based on global assumptions such as the number of man-months involved in the development or testing process and the number of problems to be solved at certain rate. Meanwhile, the empirical models are that models which uses data collected from previous developed projects to evaluate recent projects and evolve a new formulas for the current data that available [8].

Soft Computing (SC) techniques is a relatively new concept which was first defined in 1994 [9]. Neurocomputing and fuzzy logic and hybridized version of both the neuro-fuzzy were first presented. The domain of SC was expanded to cover techniques such as Genetic Algorithms (GAs), Swarm Intelligence (SI), Differential Evolution (DE) and many others. Two innovative model structures were proposed to estimate software projects effort inspired from the Constructive Cost Model (COCOMO) using Genetic Algorithms (GAs) were presented in [10]. The developed models were tested on NASA software project data set with promising results. In [11], a multiple linear based fuzzy models were proposed to model the effort and function points. The proposed fuzzy models show better estimation capabilities compared to other reported models in the literature. An extended work on evolving software effort estimation models Using Multigene Symbolic Regression Genetic Programming was presented in [12]. In [13], authors provided a detailed study on using Genetic Programming (GP), Neural Network (NN) and Linear Regression (LR) in solving the software project estimation. A study on using soft computing techniques for the development of software effort and schedule estimation models were presented in [14].

In this paper, we plan to develop software effort estimation model based on three methods. They are regression, support vector machine and artificial neural networks. Regression methods works by developing a simple linear model for the relationship between the inputs and output variables. Least square estimation is used to estimate model parameters. SVM works by finding the best separation hyperplane which separate classes of data. It generates lines of separations. Multilayer Feedforward (MLFF) neural networks adopting the Backpropagation (BP) learning algorithm to train the network and generate the relationships between the inputs and output is a nonlinear manner. ANN can produce complex relationships based on the learning process of weight selections. We explore number of case studies in this paper to estimate the effort and function points. First: modeling the effort required for software projects as a function of source line of code and methodology. Second: modeling the function points.

978-1-5090-0478-2/15/$31.00 ©2015 IEEE

The paper is organized as follows. In Section II, we provide an example of the well-known effort and FP estimation models. In Section III, we summaries the steps for both effort and function point modeling. Number of subsections discusses the modeling process along with description of the three proposed methods. The evaluation criterion adopted in this study are given in Section V. Experimental results and discussion are presented in Section VI. Finally, we provide conclusion and future work.

II. Effort and FP Estimation Models

Most software engineers realized the essential importance of developing correct effort/cost estimates of software projects since it plays a critical role in the success of administration of software resource. In the past, many studies [15], [16] explained why it is hard to move these theoretical effort estimation model to practical use. In [15], reports the results of a survey which included 598 German software companies from 112 organizations which found from only 50% captured data on completed projects, a 14% made any attempt to generate any formal models from these data. Recently, it was reported that well-known software estimation techniques are often very inaccurate and suffer many problems [17] such as the existence of noisy data.

In the following sections, we provide a literature review on a famous algorithm model called COCOMO [18], [19]. This model is usually expressed as a function of software size. We will also describe the Albrecht model for function point estimation [20]. Albrecht originally proposed four function types [20]: files, inputs, outputs and inquiries with one set of associated weights and ten General System Characteristics (GSC).

A. COCOMO Effort Model

COnstructive COst Model (COCOMO) is one of the most famous software effort estimation model used in the literature. This model was originally developed by Barry Boehm [18], [19] and was extensively revised in [21]. COCOMO utilizes the Kilo Line Of Code (KLOC) as input variable to estimate the Effort. In [18], Boehm proposed three levels of the model named: Basic COCOMO, Intermediate COCOMO and Detailed COCOMO. The general COCOMO model equation is presented by Equation 1.

$$E = \lambda \times Size^{\mu} \times EAF \tag{1}$$

where E is the effort in man-months, λ is a calibrated constant, μ is a size scale factor, $Size$ is measured by the KLOC and EAF is the effort adjustment factor from cost factor multipliers. Three types of COCOMO models are presented in Table I. They are: Organic, Semidetached and Embedded models [22] along with the values of the parameters λ and μ. In 1995 [19], Boehm proposed an advanced model structure called COCOMO II.

One essential attribute that contribute to the modeling process of the software effort found is the KLOC. Although this attribute was significantly used in the COCOMO and other known model structure but it was found not the only significant attribute [10], [23]. Recently, tuning the parameters of the COCOMO model using differential evolution to provide a better effort estimate was presented [24].

TABLE I. Basic COCOMO models

Model Name	Effort (E)	Time (T)
Organic Model	$E = 2.4(KLOC)^{1.05}$	$T = 2.5(E)^{0.38}$
Semi-Detached Model	$E = 3.0(KLOC)^{1.12}$	$T = 2.5(E)^{0.35}$
Embedded Model	$E = 3.6(KLOC)^{1.20}$	$T = 2.5(E)^{0.32}$

B. Albrecht FP Model

Albrecht's function point model became popular during the 1980's and 1990's since it provided exceptional results compared to other effort estimation model that count on the KLOC as a measure of software size [25], [26]. Albrecht model took in consideration the software system functionality as the main attributes to measure the size of the system [20], [27]. A comparison between the Function Points [20], SLIM [5], COCOMO [18], and ESTIMACS models was provided in [28]. The results were formed based data set collected from fifteen completed software projects. The predictive capability of the models were used to evaluate various model performances. In 1983, Albrecht [27], proposed the expansion of the function type to a set of three weighting values (i.e. simple, average, complex) and fourteen General System Characteristics (GSCs) to compute the number of function point of a software project.

III. Software Process Modeling

In this research work, we are adopting the system modeling process presented in [29] to build a relationship for input-output model. The basic idea of modeling the dynamics between multiple inputs x variables and single output y variable can summaries as follows:

1) **Data Collection**: Collect data for previously developed projects such as the source line of code (KLOC), adopted development methodology (ME), and other characteristics of the planned software such as level of reliability or allowable reuse.
2) **Model Selection**: In this stage we have to decide the best possible model structure for the regression, ANN and SVM. For example, the order of the linear regression model, the number of neurons and number of layers for the ANN and the type of kernel function for the SVM method.
3) **Learning Process**: We need to adopt a suitable learning technique to develop the model. For example, the Backpropagation (BP) learning algorithm for ANN.
4) **Model Validation**: A set of performance measure criterion shall be used to validate the developed model's performance. For example, the value of the mean absolute error.

IV. Proposed Model Structures

In the following sections, we will provide a description of the models and techniques adopted in this research to solve the effort and function point estimation problem. They include the simple linear regression model that uses least square estimation to find the values of the model parameters. The second method is the SVM produce lines of separation between various data

sets according to their classes and finally the well-known ANN model that build a complex nonlinear function approximation model for given examples.

A. Linear Regression

The simple linear model Equation 2 can be expanded to a multivariate system of equations as follows:

$$y = \alpha_1 x_1 + \cdots + \alpha_n x_j \tag{2}$$

where x_j is the j^{th} independent variable. In this case, we need to use least square estimation (LSE) to compute the optimal values for the parameters $\alpha_1, \ldots, \alpha_j$. Thus, we have to minimize the optimization function L, which in this case can be presented as:

$$\varphi = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{\alpha_1} x_1 - \ldots \hat{\alpha_n} x_n)^2 \tag{3}$$

To get the optimal values of the parameters $\hat{\alpha_1}, \ldots, \hat{\alpha_n}$, we have to compute the differentiation for the functions:

$$\frac{\partial \varphi}{\partial \hat{\alpha_1}} = \frac{\partial \varphi}{\partial \hat{\alpha_2}} = \cdots = \frac{\partial \varphi}{\partial \hat{\alpha_j}} = 0 \tag{4}$$

To Solve the set of Equations 4, we differentiate and equate the results to zero. This way we shall produce number of equations equals to the number of unknowns α. Solving these set of equations using LSE shall produce estimate for the parameters and a model for the relationship between x and y. The values of the estimated parameters shall be sensitive to the available number of observations.

B. Support Vector Machine

SVM is a relatively new notion defined as a supervised learning method which recognize patterns to solve many problems in classification and regression analysis. SVM is capable of classifying data set with two different category. SVM build a classification model that is capable of separating the two categories by a clearly defined gap that is as wide as possible. The origin of SVM comes from Russia in the sixties [30], [31]. It is a nonlinear generalization algorithm based on statistical learning theory. SVM was successfully used to solve variety of modeling problems in early software quality prediction [32], software reliability forecasting [33], software quality prediction [34], predicting defect-prone software modules [35], software repository mining [36] and stock market prediction [37].

The learning process in SVM works as follows; assume we have a set of training observations D such that:

$$D = \{(x_i, y_i) | x_i \in R^l, y \in \{1, -1\}\}_{i=1}^{n} \tag{5}$$

where y_i is either 1 or 1, indicating the class to which the point x_i is belongs. x_i has a dimension of l. SVM works by finding the maximum-margin (i.e. best) hyperplane that separate the points having $y_i = 1$ from those having $y_i = -1$. Any hyperplane can be presented as given in Equation 6.

$$f(x) = w^T \psi(x) + b \tag{6}$$

In 1996, a new version of support vector machine which suits regression problems was presented by Vladimir N. Vapnik et all. [38]. It was named support vector regression (SVR). Figure 1 illustrates the idea of the optimal hyperplane in SVM that separates two classes. In the left part of the figure, lines separated data but with small margins while on the right an optimal line separates the data with the maximum margins. In case of the training data are linearly-separable in the feature space of $\psi(x)$, this means that the training examples are adequately well separated thus we can draw a hyperplane between them. SVM maps the training vector x_i using the function ψ in order to find many linear separator hyperplane which maximize the margin.

SVM [39], [40] require the solution of the following optimization problem:

$$\begin{array}{ll} Minimize & \sum_{i=1}^{N} \frac{1}{2} ||w||^2 \\ Subject\ to & y_i - w^T \psi(x_i) - b \leq \epsilon \end{array} \tag{7}$$

where x_i is a training sample with target value y_i. The prediction values $\hat{y_i}$ is computed using $w^T \psi(x_i) + b$. ϵ is a threshold parameter. A kernel function $K(x, y)$ is essential element for an SVM learning process. Now a day, many kernels were proposed for the SVM. Some are listed below:

- Linear: $K(x, y) = x^T y + c$
- Polynomial: $K(x, y) = (\gamma x^T y + r)^d > 0$
- Sigmoid: $K(x_i, y_i) = tanh(\gamma x_i^T x_j + r)$
- Radial Basis Function: $K(x, y) = exp(-\frac{||x-y||}{2\sigma})$
- Gaussian: $K(x, y) = exp(-\frac{||x-y||^2}{2\sigma^2})$

where γ, r, and d are kernel parameters.

C. Artificial Neural Networks

ANN consist of many processing units called neurons. Using a learning algorithm these units are capable of producing a function that map a relationship between inputs and output training examples. It was reported that ANN can solve diversity of software engineering problems in the past. An empirical validation of a neural network model for software effort estimation was presented in [41]. Two Neural Network models; Feed-Forward Neural Network (FFNN) and Radial Basis Neural Network (RBNN) for software development effort estimation were presented in [42]. A dissertation that provided adaptive estimation framework for software defect fix effort using neural networks was presented in [43]. The dissertation proposed an economical effort model for software product line testing. A review of ANN based models for software effort estimation is provided in [44]. The system modeling process based ANN is presented in Figure 2 and adopted from [29].

The proposed block diagrams for the effort and function point models are presented in Figure 3. This is the same structure adopted for the other two methods adopted (i.e. linear regression and SVM).

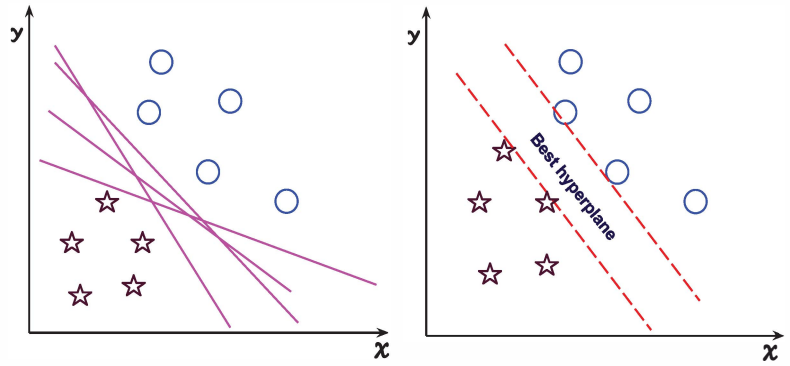

Fig. 1. Optimal hyperplane in Support Vector Machine

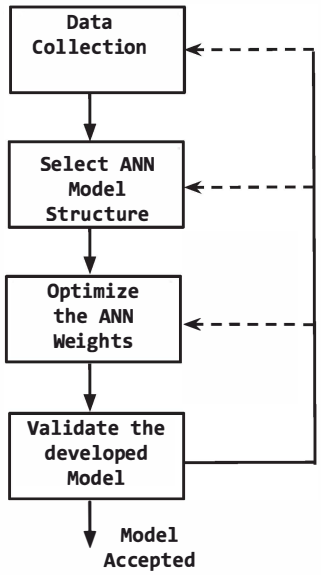

Fig. 2. System Modeling Process based ANN

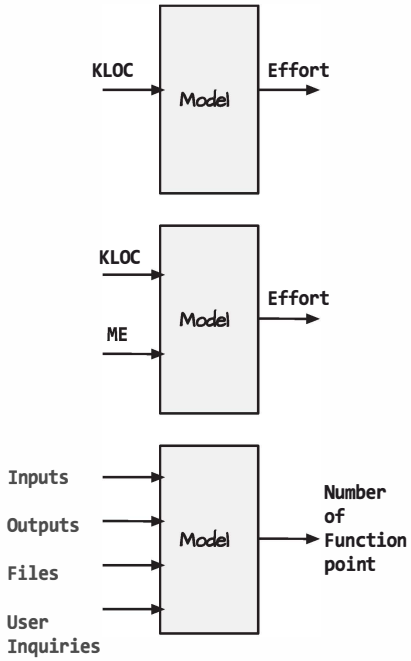

Fig. 3. Proposed structure for the effort and FP models

V. Model Validation

In order to check the performance of the developed models, we adopted number of evaluation criterion such as: the Root mean square error (RMSE), Relative absolute error (RAE) and Root relative squared error (RRSE). These evaluation criterion were provided by the well-known data mining software tool named Weka (Waikato Environment for Knowledge Analysis). It is a popular machine learning software tool written in Java, provided by the University of Waikato, New Zealand [45]. This software tool was used to develop our results. The equations which describe the evaluation criterion adopted by our models are presented as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y - \hat{y}| \tag{8}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2} \tag{9}$$

$$RAE = \frac{\sum_{i=1}^{n}|y - \hat{y}|}{\sum_{i=1}^{n}|y - \overline{y}|} \tag{10}$$

$$RRSE = \sqrt{\frac{\sum_{i=1}^{n}(y - \hat{y})^2}{\sum_{i=1}^{n}(y - \overline{y})^2}} \tag{11}$$

where y is the observed effort, $\hat{y}$ is the predicted effort and $\overline{y}$ is the mean of the effort y based n measurements.

VI. Experimental Results

A. Collected Data Set

To develop our effort estimation and FP estimation models, we used sets of data provided by [8] and [20], [27].

- **Effort Estimation Data Set 1**: The first data set used, in our experiments, was provided by Albrecht in [20], [27] and also presented in [46]. The data set is given in Table II. The data consists of 24 measurements that relates the KLOC and the development effort. The data set was sorted for experimental use. In this case, we considered the KLOC as input to the model which we are designing (See Equation 15).

$$E = f(KLOC) \tag{12}$$

- **Effort Estimation Data Set 2**: Another data set was provided by Bailey and Basili in [8] from NASA software projects. This data set consists of KLOC, Methodology (ME) and Effort for 18 software projects as shown in Table IV. In this second case, we are considering the KLOC and the ME as inputs for the model (See Equation 13). The data set was sorted for experimental use.

$$E = f(KLOC, ME) \tag{13}$$

- **FP Estimation Data Set 3**: In the FP modeling process, we adopted the Albrecht data set [20], [27] as shown in Table VII. In this case, our goal is to build a model that relates the main inputs: Inputs, Outputs, Files and Inquiries to the FP as output (See Equation 14).

$$E = f(Inputs, Outputs, Files, Inquiries) \tag{14}$$

B. Effort Models based KLOC

We developed three models for effort based the KLOC as a single input single output system. In Equation 15, we present the linear regression model. The model structure adopted is with simple order and the two parameters were estimated using least square estimation.

$$E = 0.386 \times KLOC - 1.7099 \tag{15}$$

SVM model always combine some kernel for tuning. In our experiments we adopted the Polynomial Kernel. The Polynomial kernel is a non-stationary kernel. An ANN model for the effort was also developed using MLFF neural network with single input attribute which is KLOC and a single output which is the effort. This structure looks similar to the COCOMO model where the KLOC was the only input for the model.

In Figure 4, we show the observed and predicted effort using regression, SVM and ANN. In Table II, we show the values of the computed effort in the three cases. The performance of KLOC based models is shown in Table III. ANN was able to provide the highest correlation coefficient and lowest mean absolute error. Thus, ANN outperform the other two models.

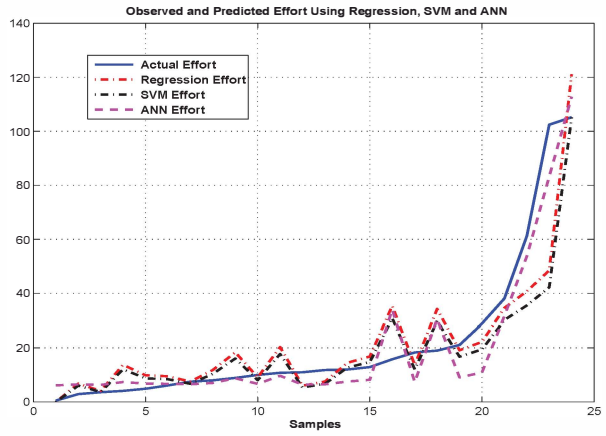

Fig. 4. Observed and Predicted Effort Using Regression, SVM and ANN

TABLE II. Computed Effort based KLOC

No.	KLOC	Effort	Regression	SVM	ANN
1	3	0.5	0	0	6.141
2	22	2.9	6.783	6.047	6.471
3	15	3.6	4.081	3.705	6.306
4	40	4.1	13.732	12.07	7.38
5	30	4.9	9.871	8.724	6.763
6	29	6.1	9.485	8.389	6.718
7	24	7.5	7.555	6.716	6.532
8	35	8	11.802	10.397	7.027
9	52	8.9	18.364	16.086	8.773
10	28	10	9.099	8.055	6.677
11	57	10.8	20.294	17.759	9.693
12	20	11	6.011	5.378	6.417
13	24	11.8	7.555	6.716	6.532
14	42	12	14.504	12.74	7.551
15	48	12.9	16.82	14.747	8.201
16	96	15.8	35.35	30.809	33.978
17	40	18.3	13.732	12.07	7.38
18	93	19	34.192	29.805	30.495
19	54	21.1	19.136	16.755	9.111
20	62	28.8	22.225	19.432	10.901
21	94	38.1	34.578	30.14	31.621
22	110	61.2	40.755	35.494	53.708
23	130	102.4	48.476	42.186	83.213
24	318	105.2	121.051	105.095	112.753

C. Effort Models based KLOC and ME

Three software effort estimation models based regression, SVM and ANN were developed based on the KLOC and ME. We adopted the linear regression model of order one. We used ANN with one hidden layer, learning rate of 0.3 and momentum of 0.2. The SVM model also has the polynomial kernel. In Figure 5, we show the observed and predicted effort using regression, SVM and ANN models.

In Table IV, we show the values of the computed effort in the three cases. The evaluated performance criterion for KLOC

TABLE III. Performance of KLOC based models

Criterion	Regression	SVM	ANN
Correlation coefficient	0.8651	0.8651	0.9543
Mean absolute error	8.5154	7.9355	6.432
Root mean squared error	13.9574	14.5816	8.516
Relative absolute error	45.1456%	42.0709%	34.1002%
Root relative squared error	50.1681%	52.4117%	30.6099%
Total Number of Instances	24	24	24

TABLE IV. COMPUTED EFFORT BASED KLOC AND ME

No	KLOC	ME	Effort	Regression	SVM	ANN
1	2.1	28	5	4.496	5.099	3.956
2	3.1	26	7	8.826	8.9	6.897
3	7.8	31	7.3	8.211	9.627	8.223
4	5	29	8.4	7.153	8.029	6.556
5	4.2	19	9	20.578	18.796	14.922
6	10.5	34	10.3	7.672	9.892	8.88
7	9.7	27	15.6	16.729	17.088	14.501
8	12.8	26	18.9	22.581	22.683	19.855
9	12.5	27	23.9	20.7	21.067	18.39
10	21.5	31	28.5	27.638	29.093	27.378
11	31.1	35	39.6	35.427	37.973	37.911
12	46.5	19	79	80.561	78.901	81.281
13	54.5	20	90.8	90.45	89.078	91.447
14	46.2	20	96	78.68	77.285	79.764
15	67.5	29	98.4	95.78	96.837	99.513
16	78.6	35	98.7	102.784	105.466	106.881
17	90.2	30	115.8	126.513	127.901	119.696
18	100.8	34	138.3	135.72	138.201	124.384

and ME based models is shown in Table V. In this case, the correlation coefficient and the mean absolute error are almost the same with slight improvement for the ANN model.

$$E = 1.418 \times KLOC - 1.456 \times ME + 42.287 \qquad (16)$$

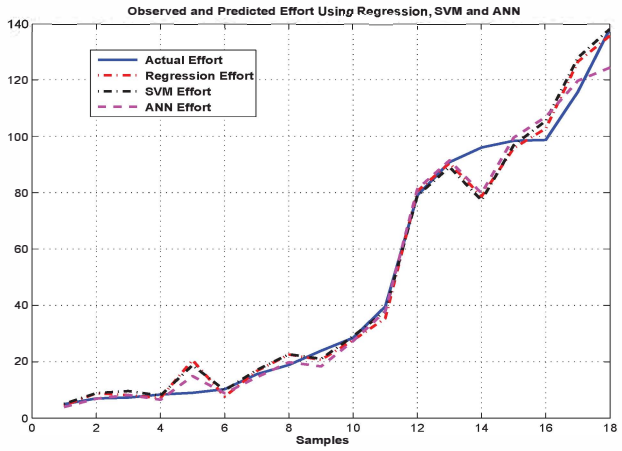

Fig. 5. Observed and Predicted Effort Using Regression, SVM and ANN

TABLE V. PERFORMANCE OF THE KLOC-ME BASED MODELS

Criterion	Regression	SVM	ANN
Correlation coefficient	0.991	0.9905	0.9916
Mean absolute error	3.9428	3.6829	3.7723
Root mean squared error	5.9521	6.1499	5.8939
Relative absolute error	9.5725%	8.9416%	9.1586%
Root relative squared error	13.3942%	13.8394%	13.2632%
Total Number of Instances	18	18	18

D. Function Point Models

We developed models for the function point using regression, SVM and ANN. Four attribute were used as inputs for the proposed model. They are: Inputs, Outputs, Files and Inquiries. The linear model developed for the function point is presented in Equation 17.

$$\begin{aligned} FP &= 3.9117 \times Inputs + 6.4326 \times Outputs \\ &+ 9.8716 \times Files + 3.4095 \times Inquiries \\ &- 47.1927 \end{aligned} \qquad (17)$$

Figure 6 shows the observed and predicted FP using ANN. The developed linear regression, SVM and ANN performance are computed and reported in Table VI. The observed and predicted values of the FP based the ANN model is given in Table VII. The SVM model provided the least mean absolute error with a correlation coefficient almost the same for the three developed models.

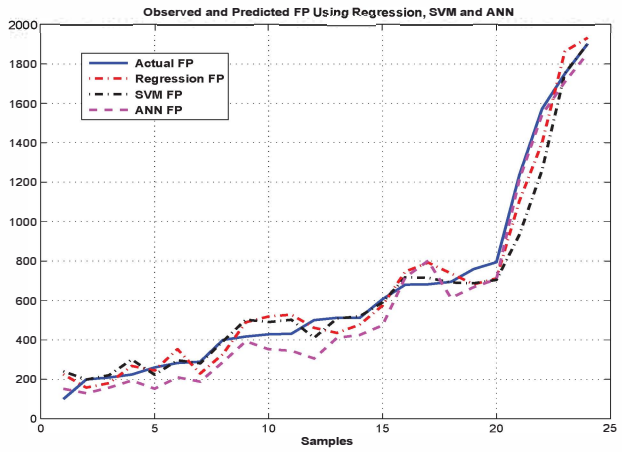

Fig. 6. Observed and Predicted FP models using Regression, SVM and ANN

TABLE VI. PERFORMANCE OF FUNCTION POINT MODELS

Criterion	Regression	SVM	ANN
Correlation coefficient	0.9857	0.9801	0.992
Mean absolute error	71.8661	61.6962	77.3947
Root mean squared error	81.2526	103.27	86.9402
Relative absolute error	20.1553	17.303	21.7058
Root relative squared error	16.8499	21.4158	18.0294
Total Number of Instances	24	24	24

VII. CONCLUSIONS AND FUTURE WORK

In this paper we studied the problem of software effort and function point estimation for software projects. This was reported as a challenging problem for software engineering community. We adopted number of methods to handle this problem; linear regression, support vector machine and artificial neural networks. The developed models based adopted methods were tested using data set from real software projects. It was found that ANN effort estimation models are more accurate with respect to error computation. For the function points models, the results for the three models were almost similar with slight improvement in the mean absolute error in the SVM model. This work will be extended by exploring other forms of soft computing techniques to estimate both the effort and the number of function points for software engineering projects.

TABLE VII. COMPUTED FUNCTION POINTS USING VARIOUS METHOD

No.	Inputs	Outputs	Files	Inquiries	FP	Regression	SVM	ANN
1	34	14	5	0	100	225.22	240.373	151.873
2	15	15	3	6	199	158.044	198.744	129.328
3	7	12	8	13	209	180.678	220.915	157.309
4	33	17	5	8	224	267.883	300.144	193.754
5	12	15	15	0	260	244.311	223.009	152.302
6	13	19	23	0	283	352.926	296.626	209.736
7	17	17	5	15	289	229.162	281.152	188.092
8	27	20	6	24	400	328.134	393.181	287.602
9	28	41	11	16	417	489.213	502.715	393.109
10	70	27	12	0	428	518.767	490.237	352.4
11	10	69	9	1	431	528.03	501.92	345.085
12	25	28	22	4	500	461.527	410.755	305.647
13	41	27	5	29	512	435.103	508.767	410.092
14	28	38	9	24	512	477.448	519.496	424.79
15	42	57	5	12	606	574.031	590.101	473.451
16	45	64	16	14	680	746.201	717.585	717.584
17	43	40	35	20	682	792.012	714.984	801.141
18	61	68	11	0	694	737.428	690.652	612.934
19	40	60	12	20	759	681.883	686.902	667.259
20	40	60	15	20	794	711.498	704.358	708.114
21	48	66	50	13	1235	1103.026	932.456	1211.953
22	69	112	39	21	1572	1399.762	1260.829	1542.806
23	25	150	60	75	1750	1863.505	1752.364	1708.501
24	193	98	36	70	1902	1932.209	1901.744	1852.864

REFERENCES

[1] C. F. Kemere, "An empirical validation of software cost estimation models," *Communication ACM*, vol. 30, pp. 416–429, 1987.

[2] J. W. Park R, W. Goethert, "Software cost and schedule estimating: A process improvement initiative," tech. rep., 1994.

[3] M. Boraso, C. Montangero, and H. Sedehi, "Software cost estimation: An experimental study of model performances," tech. rep., 1996.

[4] K. Pillai and S. Nair, "A model for software development effort and cost estimation," *IEEE Trans. on Software Engineering*, vol. 23, pp. 485–497, 1997.

[5] L. Putnam, "A general empirical solution to the macro software sizing and estimation problem," *IEEE Transactionson Software Engineering*, vol. 4, no. 4, pp. 345–381, 1978.

[6] L. Putnam and W. Myers, *Measures for excellence*. Yourdon Press Computing Series, 1992.

[7] C. E. Walston and C. P. Felix, "A method of programming measurement and estimation," *IBM Syst. J.*, vol. 16, pp. 54–73, Mar. 1977.

[8] J. W. Bailey and V. R. Basili, "A meta-model for software development resource expenditures," in *Proceedings of the 5th International Conference on Software Engineering*, ICSE '81, (Piscataway, NJ, USA), pp. 107–116, IEEE Press, 1981.

[9] L. A. Zadeh, "Soft computing and fuzzy logic," *IEEE Softw.*, vol. 11, pp. 48–56, Nov. 1994.

[10] A. Sheta, "Estimation of the COCOMO model parameters using genetic algorithms for nasa software projects," *Journal of Computer Science*, vol. 2, pp. 118–123, 2006.

[11] A. F. Sheta and S. Aljahdali, "Software effort estimation inspired by COCOMO and FP models: A fuzzy logic approach," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 11, pp. 192–197, 2013.

[12] S. Aljahdali and A. F. Sheta, "Evolving software effort estimation models using multigene symbolic regression genetic programming," *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 12, pp. 52–57, 2013.

[13] J. J. Dolado and L. F. andez, "Genetic programming, neural network and linear regression in software project estimation," in *Proceedings of the INSPIRE III, Process Improvement through training and education*, pp. 157–171, British Company Society, 1998.

[14] A. Sheta, D. Rine, and A. Ayesh, "Development of software effort and schedule estimation models using soft computing techniques," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE CEC 2008) within the 2008 IEEE World Congress on Computational Intelligence (WCCI 2008), Hong Kong*, pp. 1283–1289, June 2008.

[15] F. Heemstra, "Software cost estimation," *Information and Software Technology*, vol. 34, no. 10, pp. 627–639, 1992.

[16] A. L. Lederer and J. Prasad, "Information systems software cost estimating: A current assessment," *v*, vol. 8, pp. 22–33, 1993.

[17] T. R. Benala, S. Dehuri, and R. Mall, "Computational intelligence in software cost estimation: An emerging paradigm," *SIGSOFT Softw. Eng. Notes*, vol. 37, pp. 1–7, May 2012.

[18] B. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ, Prentice-Hall, 1981.

[19] B. Boehm, *Cost Models for Future Software Life Cycle Process: COCOMO2*. Annals of Software Engineering, 1995.

[20] A. J. Albrecht, "Measuring application development productivity," in *Proceedings of the Joint SHARE, GUIDE, and IBM Application Developments Symposium*, pp. 83–92, 1979.

[21] B. Boehm and et all, *Software Cost Estimation with COCOMO II*. Prentice Hall PTR, 2000.

[22] O. Benediktsson, D. Dalcher, K. Reed, and M. Woodman, "COCOMO based effort estimation for iterative and incremental software development," *Software Quality Journal*, vol. 11, pp. 265–281, 2003.

[23] A. Sheta, "Software effort estimation and stock market prediction using takagi-sugeno fuzzy models," in *Proceedings of the 2006 IEEE Fuzzy Logic Conference, Sheraton, Vancouver Wall Centre, Vancouver, BC, Canada, July 16-21*, pp. 579–586, 2006.

[24] S. Aljahdali and A. Sheta, "Software effort estimation by tuning COOCMO model parameters using differential evolution," in *2010 IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, pp. 1–6, 2010.

[25] R. Rask, P. Laamanen, and K. Lyytinen, "A comparison of albrecht's function point and symons' mark II metrics," in *Proceedings of the thirteenth international conference on Information systems*, ICIS '92, (Minneapolis, MN, USA), pp. 207–221, University of Minnesota, 1992.

[26] S. Furey, "Why we should use function points [software metrics]," *IEEE Softw.*, vol. 14, pp. 28, 30–, Mar. 1997.

[27] A. J. Albrecht and J. E. Gaffney, "Software function, source lines of code, and development effort prediction: A software science validation," *IEEE Transactions on Software Engineering*, vol. 9, no. 6, pp. 639–648, 1983.

[28] C. F. Kemerer, "An empirical validation of software cost estimation models," *Commun. ACM*, vol. 30, pp. 416–429, May 1987.

[29] L.Ljung, *System Identification - Theory For The User*. Prentice Hall, 1999.

[30] V. Vapnik and A. Lerner, "Pattern recognition using generalized portrait method," *Automation and Remote Control*, vol. 24, pp. 774–780, 1963.

[31] V. N. Vapnik and A. Y. Chervonenkis, "A class of algorithms for pattern recognition learning," *Avtomat. i Telemekh.*, vol. 25, no. 6, p. 937945, 1964.

[32] F. Xing, P. Guo, and M. R. Lyu, "A novel method for early software quality prediction based on support vector machine," in *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering*, ISSRE '05, (Washington, DC, USA), pp. 213–222, IEEE Computer Society, 2005.

[33] P.-F. Pai and W.-C. Hong, "Software reliability forecasting by support vector machines with simulated annealing algorithms," *J. Syst. Softw.*, vol. 79, pp. 747–755, June 2006.

[34] X. Jin, Z. Liu, R. Bie, G. Zhao, and J. Ma, "Support vector machines for regression and applications to software quality prediction," in *Proceedings of the 6th International Conference on Computational Science - Volume Part IV*, ICCS'06, (Berlin, Heidelberg), pp. 781–788, Springer-Verlag, 2006.

[35] K. O. Elish and M. O. Elish, "Predicting defect-prone software modules using support vector machines," *J. Syst. Softw.*, vol. 81, pp. 649–660, May 2008.

[36] L. Yu, "An evolutionary programming based asymmetric weighted least squares support vector machine ensemble learning methodology for software repository mining," *Inf. Sci.*, vol. 191, pp. 31–46, May 2012.

[37] A. Sheta, S. Ahmed, and H. Faris, "A comparison between regression, artificial neural networks and support vector machines for predicting stock market index," *International Journal of Advanced Research in Artificial Intelligence (IJARAI)*, vol. 4, no. 7, pp. 55–63, 2009.

[38] H. Drucker, C. J. Burges, L. Kaufman, C. J. C, Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," 1996.

[39] B. E. Boser and et al., "A training algorithm for optimal margin classifiers," in *In Proceedings of the 5 th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152, ACM Press, 1992.

[40] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, pp. 273–297, Sept. 1995.

[41] H. Park and S. Baek, "An empirical validation of a neural network model for software effort estimation," *Expert Syst. Appl.*, vol. 35, pp. 929–937, Oct. 2008.

[42] V. S. Dave and K. Dutta, "Comparison of regression model, feed-forward neural network and radial basis neural network for software development effort estimation," *SIGSOFT Softw. Eng. Notes*, vol. 36, pp. 1–5, Sept. 2011.

[43] H. Zeng, *Adaptive Estimation Framework for Software Defect Fix Effort Using Neural Networks*. PhD thesis, Fairfax, VA, USA, 2005.

[44] V. S. Dave and K. Dutta, "Neural network based models for software effort estimation: A review," *Artif. Intell. Rev.*, vol. 42, pp. 295–307, Aug. 2014.

[45] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Exploration Newsletter*, vol. 11, pp. 10–18, 2009.

[46] C. Schofield, *An Empirical Investigation into Software Effort Estimation by Analogy*. PhD thesis, Bournemouth University, 1998.