V. Resmi* and S. Vijayalakshmi

Analogy-Based Approaches to Improve Software Project Effort Estimation Accuracy

https://doi.org/10.1515/jisys-2019-0023
Received January 15, 2019; previously published online June 27, 2019.

Abstract: In the discipline of software development, effort estimation renders a pivotal role. For the successful development of the project, an unambiguous estimation is necessitated. But there is the inadequacy of standard methods for estimating an effort which is applicable to all projects. Hence, to procure the best way of estimating the effort becomes an indispensable need of the project manager. Mathematical models are only mediocre in performing accurate estimation. On that account, we opt for analogy-based effort estimation by means of some soft computing techniques which rely on historical effort estimation data of the successfully completed projects to estimate the effort. So in a thorough study to improve the accuracy, models are generated for the clusters of the datasets with the confidence that data within the cluster have similar properties. This paper aims mainly on the analysis of some of the techniques to improve the effort prediction accuracy. Here the research starts with analyzing the correlation coefficient of the selected datasets. Then the process moves through the analysis of classification accuracy, clustering accuracy, mean magnitude of relative error and prediction accuracy based on some machine learning methods. Finally, a bio-inspired firefly algorithm with fuzzy analogy is applied on the datasets to produce good estimation accuracy.

Keywords: Effort estimation; analogy-based estimation; classification; clustering; firefly optimization; fuzzy analogy; linear regression; multilayer perceptron; *k*-means algorithm; EM algorithm.

1 Introduction

The need for software project effort prediction has been increasing for the last 20 years. The predicted effort is used to find the overall cost and duration of the project. This prediction may lead to either underestimation or overestimation [5]. If it is over or under, it causes several problems in the business plans of the company. Especially, it causes several budgeting problems and schedule slippage [24].

The first thought of software effort estimation came with the presentation of the rule of thumb [13] during the 1950s. Thereafter in the 1960s, a new approach for software effort estimation was unveiled as the consequence of an expert judgment where domain experts applied their prior experiences to discern the effort of the new project [22]. The existing representations on linear equations and regression analysis were proposed [6] in 1965. The first automated tool for effort estimation was Interactive Productivity and Quality [13] established by the IBM researchers. Subsequently, Barry Boehm put forward a new mathematical model based on the regression analysis named COCOMO (COst COnstructive MOdel). This model predicts the software project effort based on the type of project. Ultimately, he propounded another model named COCOMO II which was an augmented version of COCOMO [5]. Furthermore, the models such as Putnam's Software Lifecycle Management [24], Software Evaluation and Estimation of Resources – Software Estimating Model [6] and Function Point (FP) by Albrecht were also used for effort prediction [1]. Analogy-based estimation (ABE) was fostered in the year 1997 [27] as a comparative method.

***Corresponding author: V. Resmi,** Department of Computer Applications, Udaya School of Engineering, Vellamodi, Tamil Nadu, 629 204, India, e-mail: vreshmi15@gmail.com
S. Vijayalakshmi: Department of Computer Applications, Thiagarajar College of Engineering, Madurai, Tamil Nadu, 625 015, India

Estimation by analogy is one form of expert judgment and it is also known as top-down estimating which mainly determines the duration to finish the project. Analogous estimating uses similar past projects' historical data to estimate the duration or cost of a current project, thus the term used analogy.

ABE put the estimated project alongside with the already completed projects on the basis of a measure, as an uncomplicated process. It distinguished adjacent analogies on account of similarity [3, 16, 20]. It deploys distance measures. Distance measure bestows how closely one project is tantamount to other projects. Each of the attributes' values taken for effort estimation is applied to the distance measure to descry how contiguous one object is to another. Henceforth, similar data objects are aggregated. Under the aegis of similar data objects, software project effort is estimated [10, 25].

The machine learning method of estimation has been popular for the last two decades, because machine–learning-based estimation gives more accurate results when compared with the previous two methods [12]. The machine learning method uses artificial intelligence-based techniques to give better results.

The software project effort estimation was really complicated throughout the rudimentary stages of software development. To provide more veracious results, the experience of the erstwhile project effort estimation attributes is taken into consideration. On these attributes, mining techniques are adapted to procure the effort prediction for the current project.

Predominantly, data mining bestows as a method to turn raw data into profitable and intelligent information. It has numerate functionalities [10], one among which is clustering. Clustering pertains to the grouping of data objects. Clustering follows unsupervised learning where class labels are not used. Preferably, it procreates labels for data objects. The objects are grouped or clustered based on the principle of minimizing the interclass similarity and maximizing the intraclass similarity. Particularly, all objects of that cluster are similar once it is formed. But data objects from other clusters are dissimilar. Clustering is otherwise known as data segmentation because clustering allocates large datasets into groups on the basis of their similarity [25]. Here how clusters of different methods improve the accuracy of the effort estimation is the main core of this paper.

2 Related Work

Estimation based on analogy compares the estimated project with the already completed projects based on some measures. Here the measurement is mostly distance measures. The distance measure is used to find how closely one project is related to other projects. In the initial stages of software development, software project effort estimation is very difficult. To get the more accurate results, the experience of the previous projects' effort estimation attributes is taken into consideration. On these, attributes mining techniques are applied to get the effort prediction for the current project.

Scarcely there is any model which estimates the software project effort for all domains and all kinds of applications. It is on the basis of existing models that the new models are proposed. To bring forth the effort of the new one, analogy-based estimation confronts the completed projects. Eventually, Khatibi et al. [18] contemplated a novel idea of a framework to combine analogy-based effort estimation and neural networks to ameliorate the accuracy of effort prediction. Then Humayun and Gang [12] assured that machine learning methods give us more accurate effort estimation as compared to the traditional methods of effort estimation.

Malathi and Sridhar [21] proposed an approach based on fuzzy logic, linguistic quantifiers and analogy-based reasoning. Their main aim was to enhance the performance of the effort estimation in software projects while dealing with numerical and categorical data. Azzeh and Nassif [4] together proposed a new method to discover the most prudent set of analogies from dataset characteristics to support the different size of datasets that have a lot of categorical features. Also, Prabhakar and Dutta [23] advocated a comparative study on artificial neural network (ANN) and support vector machine for predicting the software effort.

Araujo et al. [2] presented a multilayer dilation-erosion-linear perceptron (MDELP) model to solve problems in effort estimation. They used hybrid morphological operator and a linear operator to solve problems. Kaushik et al. [15] combined fuzzy inference system and cuckoo optimization (COA-FIS) for showing improved accuracy in software cost estimation.

The performance of the cluster subtrees is better than cluster supertrees as congenial to the studies of Kocaguneli et al. [19]. The performance of the analogy-based effort estimation can be improved by selecting project data from regions with small variance.

The hybrid method was planned by Khatibi et al. [17] to reduce the inconsistent project which leads to attaining higher accuracy for effort estimation. The similar projects were obtained in the different clusters through the C-means clustering technique. These clusters comprise the reliable and appropriate projects to estimate the development effort which are suitable to be employed by the ABE and ANN methods. The Fuzzy-class point (FCP) approach was intended by Satapathy et al. [26] for evaluating the cost of different software projects. In order to attain better accuracy, the FCP approach employs the various adaptive regression techniques for effort estimation.

Borandag et al. [7] prepared a case study for the software size estimation through MK II FPA (MK II Function Point Analysis) and FP methods. They used MK II FPA and FP methods to estimate the size of the software product. They implemented the same software by different developers to study their size estimation process and the size of the developed software is compared.

Yücalar et al. [30] developed a new multiple linear regression analysis-based effort estimation method. They used the datasets of the 10 software projects developed by 4 well-established software companies in Turkey. The results of the proposed method were compared with the standard Use Case Point method and simple linear regression-based effort estimation method.

3 Proposed Work

Finding the accurate effort for the new software project based on the historical dataset is a burden for project managers as there is no such model to estimate the effort directly. They have to think in many ways to reach an appropriate estimation. Our work concentrates on how to improve the accuracy and also on which techniques and datasets, approaches yield good results. Cocomo81, Cocomonasa60, Cocomonasa93, Deshnaris, ALBRECHT, Kemerer, Miyazaki1 and MAXWELL datasets are selected for our analysis. Among those datasets, ALBRECHT and Kemerer are based on FPA. We proposed four steps to reach good estimation accuracy:

(a) Select the classifier.
(b) Find the best clusters by applying the selected classifier from the first step.
(c) Perform analogy and optimization together to reach optimal solutions using best clusters.
(d) Find the new effort with the help of optimal solutions.

The diagram in Figure 1 shows the model of our proposed work. It also presents the four steps in arriving better estimation accuracy.

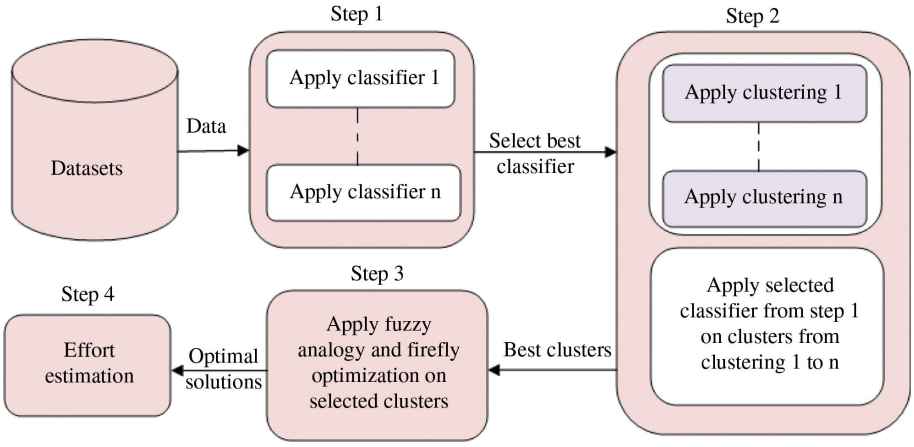

Figure 1: Proposed Learning Approach Model.

3.1 Select the Classifier

Here we have applied two classifiers on the selected datasets: multivariate linear regression and deep structured multilayer perceptron.

3.1.1 Multivariate Linear Regression

Linear regression [10] follows the equation of the line where slope, x-coordinate, y-coordinate and constant are replaced by weight, one or more independent variables, predictor variable (y) and regression coefficient. It takes the form of

$$y = b + wx, \tag{1}$$

where b and w are regression coefficients. b is the Y-intercept and w is the slope of the line. These coefficients can be thought of as weights. So the above expression (1) can be rewritten as follows:

$$y = w_0 + w_1 x. \tag{2}$$

Let D be the training set of tupels that contains $|D|$ datasets of the form (x_1, y_1), (x_2, y_2), …, $(x_{|D|}, y_{|D|})$. The regression coefficients (15) can be calculated using the following equations:

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})}, \tag{3}$$

$$w_0 = \bar{y} - w_1 \bar{x}, \tag{4}$$

where $\bar{x}$ is the mean of x and $\bar{y}$ is the mean of y.

Simple linear regression is based on only one explanatory variable. The next form of simple linear regression is multiple linear regression [6] which is based on more than one explanatory variable. Another very interesting form of linear regression is multivariate linear regression which relies on more than one predictor variable. For example, it can give more than one predictor. We have used multivariate linear regression for our work with the hope that in future there may be more than one predictor variables.

3.1.2 Deep Structured Multilayer Perceptron

A multilayer perceptron (MLP) is a feed-forward ANN model. It shows an association between given data and appropriate results. This model is represented as a directed graph which shows a set of nodes as each layer, thereby forming multiple layers. Each layer is fully connected to the next layer. The set of nodes in one layer where the input data are given is said to be the input layer. One or more nodes can be used to generate or predict output. Such nodes are in the output layer. The nodes in between the input layer and the output layer form a hidden layer. Except input nodes, all the other nodes use an activation function to generate the output. Input nodes accept only input data and just pass data to the next layer. All the other nodes use a nonlinear activation function [11] to reach the output. It is based on a supervised learning technique called backpropagation for training the network where the errors propagated in the backward direction till appropriate outputs are produced. In deep learning, each node is analyzed for more different parameters.

There are three layers in this model: an input layer, a hidden layer and an output layer. The data are given in the input layer. For all input attributes, there are nodes in the input layer. The nodes where the output is produced are in the output layer. The number of output nodes is to represent the number of classes. The nodes in between the input layer and the output layer are in the hidden layers. The link between nodes has a weight (a number) w and each node performs a weighted sum of its inputs and thresholds the result with the help of the activation function.

Figure 2 [9] shows the MLP neural network.

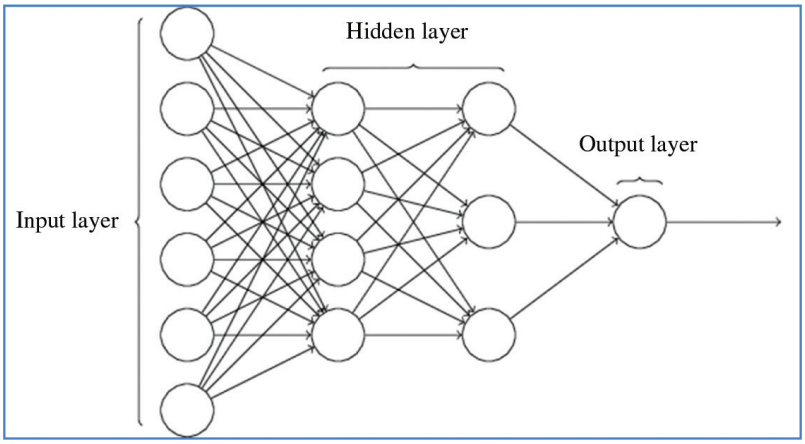

Figure 2: An Example of the Multilayer Perceptron Neural Network.

Of the two classifiers, deep structured multilayer perceptron yields good results and it is selected for the next step where this selected classifier is applied to clusters. In our work, we modified each node in a way that it learns itself more and more properties for estimation.

3.2 Select the Best Clustering Technique

Two types of clustering techniques are analyzed. They are vector quantized k-means clustering and Probabilistic Model-Based Expectation-Maximization (EM) clustering.

3.2.1 Vector Quantized k-Means Clustering

The k-means clustering method is the simplest form of clustering [10, 25]. By exercising a partitioning algorithm, it organizes the data into groups. It also splits dataset D of n objects into k partitions or k-clusters, C_1, C_2, $C_3, \ldots, C_k$, where $C_i \subset D$ and $C_i \cap C_j = \emptyset$ for $1 \leq i, j \leq k$. Each of the clusters is represented by its centroid. The centroid can be interpreted by the mean of the objects. And so forth this method is named k-means. In the initial case, some k objects are haphazardly chosen as the cluster center (centroid). Each object of every single iteration is compared with each centroid with the help of distance measures. The object which represents the lowest distance against one particular centroid belongs to the cluster of that centroid. When each iteration ends, the computed mean value for each cluster and new mean becomes the new cluster center of each cluster. And the process will reoccur till there is no change in the cluster center or the sum of squared errors between all objects in C_i and the centroid c_i is minimum for all k partitions:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} \text{dist}(p, c_i)^2. \tag{5}$$

Instead of simple k-means, we opted for vector quantized k-means to reach better results.

3.2.2 Probabilistic Model-Based Expectation-Maximization Algorithm

The EM algorithm is one of the clustering techniques which compare the given data with some mathematical model. This algorithm is an extended version of k-means clustering. In k-means clustering, each object is assigned to a particular cluster based on the distance measure, whereas in EM, each object is assigned to a cluster based on the probability of membership of the object. The EM algorithm [10] is as follows:

(a) Expectation step: each object o_i is to a cluster C_k with the probability

$$P(o_i \in C_k) = p(C_k|o_i) \tag{6}$$

$$= \frac{p(C_k)p(o_i|C_k)}{p(o_i)} \tag{7}$$

where $p(o_i \mid C_k) = N(m_k, E_k(o_i))$ follows the normal distribution around mean, m_k, with expectation E_k. This step calculates the probability of cluster membership of object x_i or the expected cluster membership of object o_i.

(b) Maximization step: re-estimate the model parameters:

$$m_k = \frac{1}{n}\sum_{i=1}^{n} \frac{o_i\, p(o_i \in C_k)}{\sum_j p(o_i \in C_j)} \tag{8}$$

This step is the "maximization" of the likelihood of the distribution of the given data.

Of the two clusters, EM cluster improves the effort estimation accuracy based on the selected classifier deep structured multilayer perceptron. So EM clusters are used in the next step to perform optimization for generating optimal solutions.

3.3 Perform Optimization

Now we have good clusters of a dataset which can improve estimation accuracy. These clusters of data are used for optimization. Here we use fuzzy analogy and firefly optimization.

3.3.1 Original Firefly Algorithm

The firefly algorithm [28, 29] was developed by Yang in the year 2009. It is a nature-inspired algorithm based on flashing light behavior of real fireflies. Each firefly is attracted to other fireflies based on the light intensity by the process of bioluminescence. Fireflies with low flashing light are attracted toward fireflies with the high flashing light. It is based on three main principles:

(1) All fireflies are unisex. Attractions of fireflies are gender independent.
(2) The attractiveness of the firefly is proportional to the brightness of the firefly, i.e. less brighter fireflies are moved to the brighter one.
(3) The brightness of the firefly is determined by the objective function.

The main feature of the firefly is its attractiveness β, and it varies with respect to distance r between fireflies. It is defined as follows:

$$\beta(r) = \beta_0 e^{-\Upsilon r^m} \tag{9}$$

where β_0 is attractiveness at $r = 0$, Υ is the light absorption coefficient and r is the distance between two fireflies x_i and x_j defined as the Cartesian distance.

$$r = |x_i - x_j| = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k}\right)^2} \tag{10}$$

where d denotes the number of dimensions.

The movement of the firefly is updated with the help of the following equation:

$$x_i = x_i + \beta_0 e^{-\Upsilon r_{i,j}^2}\left(x_j - x_i\right) + \left(\text{rand} - \frac{1}{2}\right) \tag{11}$$

where x_i is the current position of the firefly i, $\beta_0 e^{-\Upsilon r_{i,j}^2}$ is the firefly's attractiveness, α is the randomization parameter and rand is the random number generator between 0 and 1.

Firefly algorithm:

1. Define the objective function $f(x)$. $x = (x_1, x_2, \ldots, x_d)$.
2. Generate the initial population of fireflies x_i (i=1, 2, 3, …, n).
3. Set the light intensity I_i at x_i by $f(x_i)$.
4. Define the light absorption coefficient Υ.
5. While ($t <$ MaxGenerations)
   For $i = 1$ to n fireflies
      For $j = 1$ to n fireflies
         If ($I_j > I_i$)
            firefly i toward j.
            Vary attractiveness with respect to distance r via exp[Υr]
            Evaluate new solutions and update light intensity.
      End for j.
   End for i.
   Rank all fireflies and find the current global best set of fireflies.
End while

3.3.2 Fuzzy Analogy and Firefly Optimization

The fuzzy analogy is nothing but an analogy based on fuzzy logic. In analogy-based effort estimation, identical projects are identified from the historical dataset and these identified projects are used for effort estimation either by collecting opinion from the experts or by applying some mathematical model based on the similar projects. It consists of the case identification process, case retrieval process and case adaptation process.

There may be lots of instances in the historical data. So the process of finding matching cases is difficult. In the fuzzy analogy approach, all data are converted into fuzzy sets by applying fuzzy logic. Here all the variables are converted to linguistic variables by using membership functions. So categorical variables can be handled efficiently with the help of fuzzy logic. Once the fuzzy datasets are ready, our proposed work generates the fuzzy rules for the fuzzy dataset. From those fuzzy rules, optimal rules are generated with the help of the firefly optimization algorithm. In our work, three initial sets of solutions are formed with the set of flies (fuzzy rules), i.e. each solution consists of a set of flies. For each fly, the fitness value is computed. Here the fitness value is the mean magnitude of relative error (MMRE) value. For each solution, the summation of fitness values of all flies in that solution is calculated and the solution is ranked based on the minimum of that value. The solution with the minimum MMRE value is set aside and the remaining solutions are updated with other sets of rules. And this process is repeated n number of times till we get the best optimal solutions.

Once we reach optimal solutions, the next step in the fuzzy analogy is the identification of similar cases. This is achieved by finding the distance between projects p and pi_2 by comparing each individual attribute of p_1 and p_2.

The next step in the fuzzy analogy is case adaptation. In this step estimate of the new project is derived from the effort values of similar projects.

4 Results and Discussions

For assessing the performances of the k-means clustering and EM algorithm, eight datasets have been selected from the PROMISE data repository. They are Cocomo81, Cocomonasa60 and Cocomonasa93, DESHARNAIS, ALBRECHT, Kemerer, Miyazaki1 and MAXWELL datasets. Cocomo81 has 63 instances and 17 attributes (all numeric: 15 for the effort multipliers, one for Lines of Code (LOC) and one for actual development effort), and there is no missing attribute. Cocomonasa60 has 60 instances and 17 attributes (15 discrete in the range very

low to extra high). Cocomonasa93 has 93 instances and 24 attributes and DESHARNAIS has 81 instances and 12 attributes. For ALBRECHT, there are 24 instances and 8 attributes. Kemerer has 15 instances and 8 attributes. For Miyazaki1, there are 48 instances and 9 attributes. MAXWELL has 62 instances and 27 attributes. Among those datasets, ALBRECHT and Kemerer are FPA-based datasets and Miyazaki1 is a COBOL dataset.

Parameters for validation:

- Correlation coefficient: Correlation tells how much actual and predicted are related. Its value ranges from -1 to 1, where 0 is no relation, 1 is the very strong linear relation and -1 is an inverse linear relation.
- Mean magnitude of relative error (MMRE): There are many measures to predict the accuracy of the effort prediction models. But the commonly used measure is the MMRE.
  The MMRE can be measured by the following formula:

$$\text{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\text{MRE}i \tag{12}$$

  where MRE is the magnitude of relative error.

$$\text{MRE} = \frac{|\text{act}_{\text{effort}} - \text{est}_{\text{effort}}|}{|\text{act}_{\text{effort}}|} \tag{13}$$

  $\text{MMRE} \leq 0.25$ is the acceptable range [8].
- Prediction (PRED): This is also another measure to estimate the accuracy [14]:

$$\text{PRED}(0.25) = \frac{k}{n} \tag{14}$$

  where k is the number of observations whose MRE is less than or equal to 0.25 and n is the number of observations.
- Classification accuracy: This is the percentage of the ratio of the number of projects classified correctly to the total number of projects within the dataset. A high value of classification accuracy leads to good accuracy.
- Clustering accuracy: This is the percentage of the ratio of the number of projects grouped correctly to the total number of projects within the dataset. A high value of clustering accuracy leads to good accuracy.

Table 1 shows the results of the four validation parameters for multivariate linear regression effort estimation.

From Table 1, it is noted that with the higher value of correlation coefficient the better prediction yields. For the better correlation coefficient values, the classification accuracy also increases.

Table 2 shows the results of the four validation parameters for deep structured multilayer perceptron effort estimation.

From Table 2, it is also noted that higher the correlation coefficient value better the prediction and classification accuracy. But when we compare the above-mentioned two techniques based on MMRE and prediction,

Table 1: Validation Parameters' Values for Multivariate Linear Regression Effort Estimation.

Datasets	Multivariate linear regression effort estimation			
	Correlation coefficient	Classification accuracy	MMRE	Prediction
Cocomo81	0.706	70.6	0.265	59.3
Cocomonasa60	0.716	71.6	0.255	60.3
CocomoNasa93	0.72.7	72.7	0.245	61.3
Desharnais	0.737	73.7	0.235	63.3
ALBRECHT FPA	0.91	75.2	0.23	61.5
Kemerer FPA	0.37	40.2	0.55	30.2
Miyazaki1 COBOL	0.05	20	0.85	22.3
MAXWELL	0.81	65.4	0.20	69.3

Table 2: Validation Parameters' Values for Deep Structured Multilayer Perceptron Effort Estimation.

Dataset	Deep structured multilayer perceptron effort estimation			
	Correlation coefficient	Classification accuracy	MMRE	Prediction
Cocomo81	0.767	70.6	0.205	66.3
Cocomonasa60	0.777	71.6	0.195	68.3
CocomoNasa93	0.787	72.7	0.185	69.3
Desharnais	0.797	73.7	0.175	71.9
ALBRECHT FPA	0.75	70.3	0.232	65.88
Kemerer FPA	0.35	34.4	0.595	30.23
Miyazaki1 COBOL	0.96	92.6	0.15	86.8
MAXWELL	0.76	73.1	0.195	69.4

deep structured multilayer perceptron effort estimation performs better. So in the next step, we use deep structured multilayer perceptron classifier as the estimation model to estimate the effort on clustered data.

In Table 3, the values of validation parameters for deep structured multilayer perceptron effort estimation using vector quantized *k*-means clusters are tabulated.

Table 4 shows the values of validation parameters for deep structured multilayer perceptron effort estimation using probabilistic model-based EM clusters.

When we compare MMRE and prediction values in Tables 3 and 4, it is found that probabilistic model-based EM clusters give good accuracy in prediction and less MMRE values than vector quantized *k*-means

Table 3: Deep Structured Multilayer Perceptron Effort Estimation Using Vector Quantized *k*-Means Clusters.

Dataset	Vector quantized *k*-means clustering and deep structured multilayer perceptron effort estimation			
	Clustering accuracy	Classification accuracy	MMRE	Prediction
Cocomo81	71.3	75.7	0.215	64
Cocomonasa60	72.9	77.2	0.205	66
CocomoNasa93	74.5	79.2	0.195	67.9
Desharnais	76.3	80.7	0.185	69.9
ALBRECHT FPA	80	75.1	0.191	69.21
Kemerer FPA	48	39.91	0.43	43.66
Miyazaki1 COBOL	98	94.72	0.13	90.1
MAXWELL	79	76.01	0.15	74.44

Table 4: Deep Structured Multilayer Perceptron Effort Estimation Using Probabilistic Model-Based Expectation-Maximization (EM) Clusters.

Dataset	Probabilistic model-based EM and deep structured multilayer perceptron effort estimation			
	Clustering accuracy	Classification accuracy	MMRE	Prediction
Cocomo81	82.2	85.3	0.165	72.9
Cocomonasa60	85	87.8	0.155	74.9
CocomoNasa93	87.6	91	0.145	75.8
Desharnais	89.7	94	0.135	77.8
ALBRECHT FPA	83	78.2	0.16	75.12
Kemerer FPA	53	42.12	0.4	48.22
Miyazaki1 COBOL	99	96.48	0.10	94.12
MAXWELL	80	76.77	0.14	78.64

clusters. So probabilistic model-based EM clusters are used by firefly optimization and fuzzy analogy for effort estimation. The result of this approach is shown in Table 5.

From Table 5, it is evident that prediction accuracy measures like MMRE and prediction values are much more improved than other previous methods used in steps 1 and 2 of the proposed method. Table 6 tabulates the performance measures of our proposed method with the other two existing methods (MDELP and COA-FIS).

Figures 3 and 4 graphically show that MMRE values of the proposed method for the four selected datasets are less than the MMRE values of the existing methods. Also prediction values increase when compared with two methods. Hence, the proposed method improves the accuracy of the effort estimation.

From our experiments, it is evident that the MMRE values of Cocomo81, Cocomonasa60, CocomoNasa93 and Desharnais datasets decreased by 62%, 63%, 66% and 68%, respectively, when compared with the existing MDELP and 49%, 51%, 53% and 56%, respectively, when compared with the existing COA-FIS. Also the MMRE values of ALBRECHT, Kemerer, Miyazaki1 and MAXWELL datasets decreased by 59%, 21%, 97% and 50%, respectively, when compared with the existing MDELP and 58%, 35%, 663% and 45%, respectively, when compared with the existing COA-FIS.

Similarly, the prediction values of Cocomo81, Cocomonasa60, CocomoNasa93 and Desharnais datasets increased by 56%, 52%, 53% and 53%, respectively, when compared with the existing MDELP and 29%, 28%, 27% and 30%, respectively, when compared with the existing COA-FIS. Also, the prediction values of ALBRECHT, Kemerer, Miyazaki1 and MAXWELL datasets increased by 55%, 24%, 73% and 44%, respectively, when compared with the existing MDELP and 37%, 18%, 47% and 21%, respectively, when compared with the existing COA-FIS.

Table 5: Effort Estimation Using Expectation-Maximization (EM) Clusters, Firefly Optimization and Fuzzy Analogy.

Dataset	Probabilistic model-based EM, firefly optimization and fuzzy analogy effort estimation			
	Clustering accuracy	Classification accuracy	MMRE	Prediction
Cocomo81	82.2	80.7	0.125	78.8
Cocomonasa60	85	81.7	0.115	79.8
CocomoNasa93	87.6	83.3	0.105	81.8
Desharnais	89.7	84.3	0.095	84.6
ALBRECHT FPA	88	81.12	0.10	85.88
Kemerer FPA	69	59.66	0.23	55.01
Miyazaki1 COBOL	99.9	98.88	0.010	97.76
MAXWELL	83	79.34	0.10	83.44

Table 6: Comparison of Proposed Work with Two Existing Methods.

Dataset	Existing MDELP		Existing COA-FIS		Proposed method	
	MMRE	Prediction	MMRE	Prediction	MMRE	Prediction
Cocomo81	0.325	50.5	0.245	61.2	0.125	78.8
Cocomonasa60	0.315	52.4	0.235	62.2	0.115	79.8
CocomoNasa93	0.305	53.4	0.225	64.2	0.105	81.8
Desharnais	0.295	55.4	0.215	65.3	0.095	84.6
ALBRECHT FPA	0.243	55.25	0.241	62.6	0.10	85.88
Kemerer FPA	0.29	44.4	0.331	46.7	0.23	55.01
Miyazaki1 COBOL	0.29	56.6	0.2	66.4	0.010	97.76
MAXWELL	0.20	58.14	0.19	69	0.10	83.44

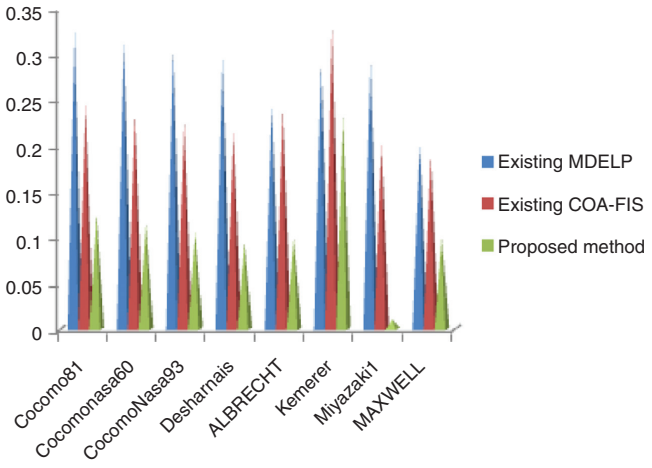

Figure 3: MMRE Comparison between the Existing and Proposed Method.

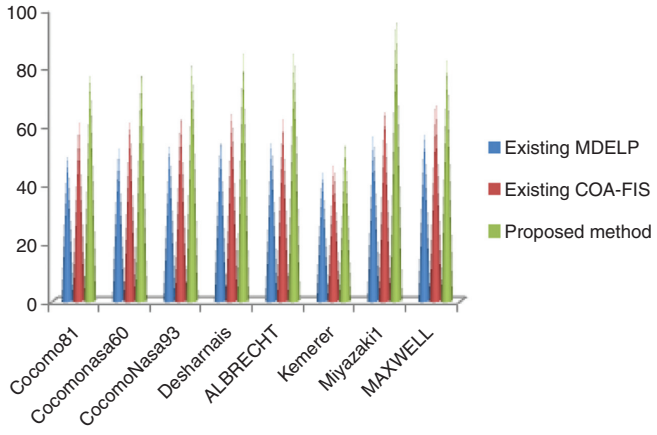

Figure 4: Prediction Comparison between the Existing and Proposed Method.

5 Conclusion

Congenial to the results of several researchers, it is incontestable that there is no approach which is suitable for estimating software project effort for all domains and all kinds of applications. Thus, it is indispensable to use the prior experiences of the projects to estimate the effort of the current project. Analogy-based effort estimation is one among them. So it can be implemented on machine learning techniques to derive better analogies. In this paper, we gave emphasis to two learning approaches such as classification and clustering. Of the two clustering methods, EM clusters are more excelling than vector quantized k-means clusters contingent on MMRE and prediction values. So EM clusters are taken for fuzzy analogy and firefly optimization to get the optimal solutions. From the optimal solutions, the effort of the new project is derived with good accuracy. Different optimization techniques have been left for future for analyzing which optimization suits for which type of domain datasets. Different clustering and classification techniques can be considered for future work. And also, it could be better if some data pre-processing techniques are applied before the data applied for clustering and classification.

Bibliography

[1] A. J. Albrecht and J. A. Gaffney, Software function, source lines of codes, and development effort prediction: a software science validation, *IEEE Trans. Softw. Engg.* **9** (1983), 639–648.

[2] R. de A. Araujo, A. L. I. Oliveira and S. Meira, A class of hybrid multilayer perceptrons for software development effort estimation problems, *Expert Syst. Appl.* **90** (2017), 1–12.

[3] M. Azzeh, A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation, *Empir. Softw. Eng.* **17** (2012), 90–127.

[4] M. Azzeh and A. B. Nassif, Analogy-based effort estimation: a new method to discover set of analogies from dataset characteristics, *IET Softw.* **9** (2015), 39–50.

[5] B. W. Boehm, *Software engineering economics*, Prentice Hall, Englewood Cliffs, NJ, 1981.

[6] B. W. Boehm and R. Valerdi, Achievements and challenges in cocomo-based software resource estimation, *IEEE Softw.* **25** (2008), 74–83.

[7] E. Borandag, F. Yucalar and S. Z. Erdogan, A case study for the software size estimation through MK II FPA and FP methods, *Int. J. Comput. Appl. Technol.* **53** (2016), 309–314.

[8] S. D. Conte, H. E. Dunsmore and V. Y. Shen, *Software engineering metrics and models*, Benjamin-Cummings Publishing, Redwood City, 1986.

[9] GitHub. https://dzone.com/articles/deep-learning-via-multilayer-perceptron-classifier.

[10] J. Han and M. Kamber, *Data mining concepts and techniques*, 2nd ed., Elsevier, Amsterdam, The Netherlands, Reprinted 2008.

[11] S. Haykin, *Neural networks: a comprehensive foundation*, 2nd ed. Prentice Hall, New York, 1998.

[12] M. Humayun and C. Gang, Estimating effort in global software development projects using machine learning techniques, *Int. J. Inform. Edu. Technol.* **2** (2012), 208–211.

[13] C. Jones, *Estimating software costs: bringing realism to estimating*, 2nd ed., McGraw-Hill, New York, 2007.

[14] M. Jørgensen, Experience with the accuracy of software maintenance task effort prediction models, *IEEE Trans. Softw. Engg.* **21** (1995), 674–681.

[15] A. Kaushik, S. Verma, H. J. Singh and G. Chhabra, Software cost optimization integrating fuzzy system and COA-cuckoo optimization algorithm, *Int. J. Syst. Assur. Eng. Manage.* **8** (2017), 1461–1471.

[16] J. Keung, B. Kitchenham and D. R. Jeffery, Analogy-X: providing statistical inference to analogy-based software cost estimation, *IEEE Trans. Softw. Engg.* **34** (2008), 471–484.

[17] B. V. Khatibi, D. N. A. Jawawi, S. Z. M. Hashim and E. Khatibi, Increasing the accuracy of software development effort estimation using projects clustering, *IET Softw.* **6** (2012), 461–473.

[18] B. V. Khatibi, D. N. A. Jawawi and E. Khatibi, Increasing the accuracy of analogy based software development effort estimation using neural networks, *Int. J. Comput. Commun. Engg.* **2** (2013), 78–81.

[19] E. Kocaguneli, T. Menzies and A. Bener, Exploiting the essential assumptions of analogy-based effort estimation, *IEEE Trans. Softw. Eng.* **38** (2012), 425–438.

[20] J. Z. Li, G. Ruhe, A. Al-Emran and M. M. Ritcher, A flexible method for software effort estimation by analogy, *Empir. Softw. Eng.* **12** (2007), 65–106.

[21] S. Malathi and S. Sridhar, Estimation of effort in software cost analysis for heterogeneous dataset using fuzzy analogy, *Int. J. Comput. Sci. Inform. Security*, **10** (2012), arXiv:1211.1136.

[22] E. A. Nelson, *Management handbook for the estimation of computer programming costs*, System Developer Corp., Santa Monica, CA, USA, 1966.

[23] Prabhakar and M. Dutta, Prediction of software effort using artificial neural network and support vector machine, *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **3** (2013), 40–46.

[24] L. H. Putnam, A general empirical solution to the macrosoftware sizing and estimating problem, *IEEE Trans. Softw. Engg.* **4** (1987), 345–361.

[25] S. K. Sarangi and V. Jaglan, Performance comparison of machine learning algorithms on integration of clustering and classification techniques, *Int. J. Emerg. Technol. Comput. Appl. Sci.* (2013) 251–257.

[26] S. M. Satapathy, M. Kumar and S. K. Rath, Fuzzy-class point approach for software effort estimation using various adaptive regression methods, *CSI Trans. ICT* **1** (2013), 367–380.

[27] M. Shepperd and C. Schofield, Estimating software project effort using analogies, *IEEE Trans. Softw. Engg.* **23** (1997), 736–743.

[28] X. S. Yang, *Nature-inspired metaheuristic algorithms*, Luniver Press, London, 2008.

[29] X. S. Yang, Firefly algorithms for multimodal optimization, in: *Stochastic algorithms: foundations and applications*, vol. 5792, pp. 169–178, SAGA. Lecture Notes in Computer Sciences, Springer, Heidelberg, Berlin, 2009.

[30] F. Yücalar, D. Kilinc, E. Borandag and A. Ozcift, Regression analysis based software effort estimation method, *Int. J. Softw. Eng. Knowledge Eng.* **26** (2016) 807–826.