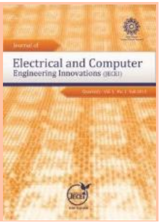
SRTTU

Journal of Electrical and Computer Engineering Innovations

JECEI, Vol. 6, No. 1, 2018

Regular Paper



A Method for Estimating the Cost of Software Using Principle Components Analysis and Data Mining

Azin Saberi Nejad[1] and Reza Tavoli[2,*]

[1]M.Sc. student of computer engineering – software, Pooyandegan Danesh Institution of Higher Education, Chalus, Iran.

[2]Assistant professor of Department of computer engineering, Islamic Azad University of chalus, Chalus, Iran.

[*]Corresponding Author's Information: r.tavoli@iauc.ac.ir

ARTICLE INFO

ARTICLE HISTORY:
Received 04 March 2018
Revised 20 July 2018
Accepted 29 July 2018

KEYWORDS:
Increased accuracy
Software cost estimation
Principle components analysis
Data mining

ABSTRACT

These days, data mining is one of the most significant issues. One field of data mining is a mixture of computer science and statistics which is considerably limited due to increase in digital data and growth of computational power of computers. One of the domains of data mining is the software cost estimation category. In this article, classifying techniques of learning algorithm of machine and COCOMO model as the most common estimation model of software costs are presented. Then, the analysis method of principal component approach is presented. This article shows that the method to increase the accuracy of software cost estimation is suitable. Moreover, the basic data set is decreased and is turned into a new collection by using this method. Among the features, the best are extracted. The algorithms of several classifications are assessed by applying this method. Finally, the evidence for accuracy of our claims in terms of increase in estimation accuracy of software costs is presented.

1. Introduction

Today, software is considered as the most expensive element of any computer system [1]. One of the domains of data mining is the cost estimation of software [2]. Software cost estimation is the process of predicting the effort required to develop a software system [3]. Much of the decision-making of managers at the start of a software project is discussion about the cost and time. The successful software project is a project to achieve certain predetermined purposes in terms of cost and time. Excessive costs for a software maker can be harmful. Cost estimation was the problem of systems analysts, project managers and software engineers for decades. Identifying the enact costs of software projects helps managers to accurately estimate the real price of a software [1].

Software projects must begin by analyzing the previous projects and those that are marketed as products. Calculation of software cost is usually tricky. Software projects were not so understandable earlier and always themes and ideas that were in customers' minds and the minds of managers indeed differed. With the gradual growth in the size and importance of applicable programs, costs of creating software began to grow and hence the excessive increase of costs for software planners were disastrous. In the previous years, various methods were presented for estimating the software project cost [1], which were called algorithmic or non-algorithmic methods.

Accurate cost estimation is important for the following reasons [4]:

- It can be used to classify and prioritize development projects with respect to complete business plan.
- It can help to find out what resources to commit to a project and how well these resources are used.

- It can help to assess the impact of changes and how to support for preplanning. Projects can be easier to manage and control when resources are better matched to real needs [4].
- Customers expect that development costs to be accurately in line with estimated costs [4].

Software cost estimation activity historically has been a major difficulty in software development. Several reasons have been identified that affects the cost estimation process such as [4]:

- Cost estimate of software development is difficult. The first steps in the estimation are to understand and define the system that the cost is to be estimated.
- A cost estimate done early in the project life cycle is generally based on less precise inputs and less detailed design specifications.
- Software development involve many inter-related factors, which affect development effort and productivity, which initially are not well understood.
- Incomplete, inaccurate or inconsistent historical database of cost measurements.
- Lack of trained estimators.
- Software is intangible, invisible, and intractable. So, it is more difficult to understand and estimate a product or process that cannot be seen or touched.

To do so, it is necessary to model data to observe the number of attempts in output by putting a related data in new projects. Therefore, the thing that helps create suitable model is using basic data set. One data set that has been considered by researchers and shows the output of different models is the data set related to NASA 93 with 93 records and 24 features. This data set is released as a result of free program of space station at 6 centers in NASA which include jet launch [5]-[6]. COCOMO data set 81 includes 63 records and 19 features. The NASA data set 93 has COCOMO data set format.

The reason why we selected these data sets are their availability. Therefore they are suitable sources to compare with other models. We also applied the principal component analysis (PCA) method which is one method to extract features. We will introduce the best collecting algorithm to increase the accuracy of software cost estimation by using PCA to decrease the input data and also to use different algorithms in classification of data mining.

This paper is formed as follows. In Section 2, related literature is discussed. Section 3 focuses on software cost estimation and related works. In Section 4, the suggested approach is presented and discussed. Section 5 focuses on experiments and results which include implementation tools, data collection, evaluation criteria, results and its analysis and finally, the conclusion and future works are presented in Section 6.

2. Related Literature

Accurately estimating software development effort is of vital importance. Under-estimation can cause schedule and budget overruns as well as project cancellation. Over-estimation delays funding to other promising ideas and organizational competitiveness [7]. The concept of software cost estimation began in 1960s and many cost estimation models have been proposed by various researchers since then [8].

Hence, there is a long history of researchers exploring software effort estimation; among these researchers are: Wolverton (1974), Black and et al. (1977), Herd and et al. (1977), Walston and Felix (1977), Freiman and Park (1979), Boehm (1981), Jensen (1983), Park (1988), Shepperd and Schofield (1997), Walkerden and Jeffery (1999), Burgess and Lefley (2001), Menzies and et al. (2006), Jorgensen and Shepperd (2007). In 2007, Jorgensen and Shepperd reported on hundreds of research papers dating back to the 1970s devoted to the topic, over half of which proposed some innovation for developing new estimation models [7]. In the 1970s and 1980s, this kind of research was focused on parametric estimation as done by Putnam and others. For example, Boehm's constructive cost model (COCOMO) model [7]. COCOMO is a parametric method; i.e., it is a model-based method that (a) assumes that the target model has a particular structure, then (b) uses model-based methods to fill in the details of a particular structure (e.g., to set some tuning parameters) [7]. Since that work on parametric estimation, researchers have innovated other methods based on regression trees (Shepperd and Schofield (1997)), case-based-reasoning (Shepperd and Schofield (1997)), spectral clustering (Menzies and et al. (2013)), genetic algorithms (Freiman and park (1979), Cordero and et al. (1997)), etc. These methods can be augmented with "meta-level" techniques like tabu search (Corazza and et al. (2010)), feature selection (Zhihao chen and et al. (2005)), instance selection (Kocaguneli and et al. (2012)), feature synthesis (Menzies and Shepperd (2012)), active learning (Kocaguneli and et al. (2013)), transfer learning (Kocaguneli and et al. (2014)). Temporal learning (Lokan and Mendes (2009), Miller (2002)), and so on [7].

3. Software Cost Estimation

Software cost estimation plays a vital role in software engineering as the success or failure of project entirely depends on it. Cost estimation's deliverables like staff requirements, schedule and effort are important chunk of information for formation and execution of a project. They provide inputs for project request and proposal, project planning, control, budget, progress monitoring & staff allocation. Illogical and uncertain estimates are the root causes of project failure. So, the capability of any system is to find out correct time and cost of software which is very crucial for the progress of that system. The software engineering community puts enormous effort for building models in order to comfort estimators to provide accurate cost estimates for software projects [9].

A. Software Cost Estimation Models

Cost estimation techniques are mainly of two kinds: algorithmic and non-algorithmic [10]-[11]-[12]. The two kinds are introduced in details.

A.1. Non-Algorithm Models

this model first compares the project under consideration with the previously done projects by the organization and analyses the information from the most similar projects to make cost estimates. Basically, this model makes use of past experiences [8]. Models explained in details are as follows:

- **Top-Down:** The top down estimation method also known as macro model, considers effort as a function of size of the project.

$$\text{Effort} = a.\ b \qquad (1)$$

where a is a constant and b is the size of the project. At first, an overall cost is estimated, the project is then partitioned into various levels and the cost estimation of every level of the project is derived from the global properties of the software project. The overall cost estimation of the project makes it very easy to estimate costs at the start, however, one needs to revise the initial estimates as the project progresses, which leads to delays if the revisions lead to varying results from the earlier estimates. Due to the fact that very little detailed information is available at the start, this method is highly regarded in early cost estimation [8].

- **Bottom-Up:** This is the exact opposite of the top-down approach. In this method, we first estimate the cost for each and every small components of the project, which is then combined to form the cost of the overall project. It aims to consolidate the small information available and how they interact in order to arrive at the overall cost. COCOMO method uses this approach for cost estimation. Although bottom-up is a much consolidated technique, but it cannot be applied to projects where much detail is not known at the start of the project. Trying to apply bottom-up in these situations can lead to bad estimations [8].

- **Analogy Model:** Cost estimating by analogy means comparing the proposed project to previously completed similar project where the project development information is known. Actual data from the completed projects are extrapolated to cost estimate the newly proposed project. Analogy method can be used either at system level or at component level. This method uses the following estimating steps [4]:

- Find out the necessary characteristics of the proposed project.
- Choose the most similar completed projects whose characteristics have been stored in a historical data base.
- Find the estimate for the proposed project from the most similar completed project by analogy.

A.2. Algorithmic Models

Algorithm models are based on one or more mathematical formulas that are typically obtained through statistical analysis. These mathematical equations are based on previous research and data and use inputs such as source code lines, a number of functions for execution, and other cost factors. Each algorithmic model is represented by Eq. (1): Effort is an action to estimate the cost, usually measured by person-month. Yi factors of cost and F is a form of the function [8], [13].

$$\text{Effort} = F(Y1, Y2, Y3, \ldots, Yn) \qquad (2)$$

- **COCOMO Model (Constructive Cost Model):** In 1981, Boehm introduced the first version of COCOMO as a model for estimating the effort, cost, and schedule. This COCOMO version was called COCOMO 81. In 1997, Boehm enhanced his first version of COCOMO and introduced another model called COCOMO II. This model provides more support for modern4 ISRN Software Engineering software development processes. In COCOMO models, LOC is used as a software code size and given in thousands to measure the effort which is measured in person-month. The basic COCOMO pattern is shown in (3). In this case, EF is the number of people - month or hours required, C is the constant value of an estimated value, LOC is the number of program lines, and K is a constant which estimated to be 1.05.

$$EF=C\ (LOC)\ K \qquad (3)$$

Variants of COCOMO models include: 1) Basic COCOMO 2) Intermediate COCOMO 3) Detailed COCOMO [8]-[9]-[13].

4. Suggested Approach

Principal components analysis is a commonly used dimensionality reduction and data analysis tool in many areas such as computer vision, data mining, biomedical informatics, and many others [14]. For years, the principal components analysis method has been considered. For example by, Pearson (1901) or Hotelling (1933); for modern reviews, Abdi & Williams (2010) or Jolliffe (2014); for uses of PCA in astronomy see e.g., Yip et al. (2004); Suzuki (2006); Conselice (2006); Budav´ari et al. (2009); P^aris et al. (2011) [15]. Another definition of the above method is in [16]-[17]-[18], which is as follows: PCA is one of the most widely used multivariate data analysis technique which was employed primarily for dimensional reduction and visualization. In this part, our purpose to increase accuracy of software cost estimations by using the decrease of input dimensions and by principal component analysis, is introduced.

The following figure shows the general processes. To do and create COCOMO data set 81 to a new data set, two software of MATLAB and rapid miner were applied. The MATLAB software resulted from software rapid miner which was changed by omitting some features due to being numerical and unsupervised of PCA, was used. In MATLAB software, this data set was changed by decreasing the dimensions and the related formulae (covariance) which resulted to a new data set. Covariance is an index to change one variable to another.

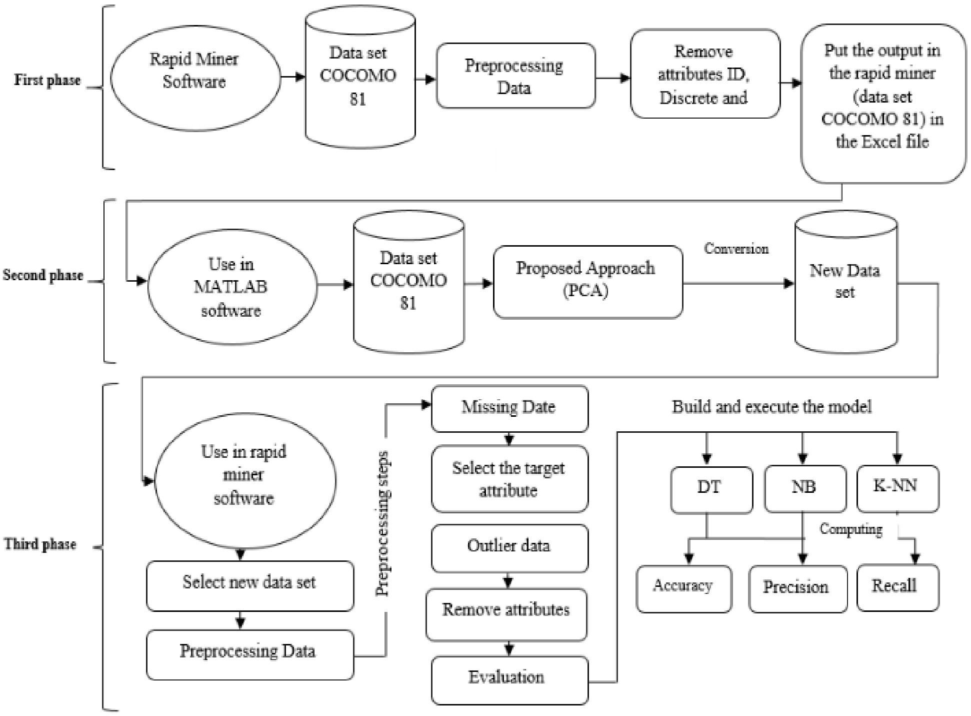


Figure 1: The general processes.

In (4) below, X is independent variable, Y is the dependent variable, i shows the number of members (or samples), $\overline{X}$ is the average of dependent variable X, $\overline{Y}$ is the average of independent variable Y, ∑ shows the collection of two parentheses and N-1 is the number of samples minus 1. (N-1 instead of N for calculating the variance of samples.)

$$\mathrm{COV}(X, Y) = \frac{\sum_{i=1}^{n}(Xi-\overline{X})(Yi-\overline{Y})}{n-1} \qquad (4)$$

According to this formula, the resulted amount if: 1) is positive, means that X or Y increase or decrease together. 2) is negative, suggests that Y decreases by increasing X or vice versa. 3) is 0, means that X and Y are independent [19]- [20]. So, new data set in rapid miner software were used to create and evaluate models by using the explained algorithms. In rapid miner, the processes are done like what is shown in the Figure 1.

To estimate the software cost and to create evaluate models, several criteria are considered and finally, the best accuracy of this method was determined using the classification technique as outlined in the next section.

5. Experiments and Results

In this part, our tests done on 2 data sets by using learning algorithm of machines and suggested methods in rapid miner software and the results are presented. In this article, classification techniques of data mining were used which will be explained later.

A. Implementation Tools

We used rapid miner in this article. Rapid miner is based on Boston, Massachusetts, U.S [21]. Rapid miner builds a software platform for data science teams that unites data prep, machine learning, and predictive model deployment. Organizations can build machine learning models and put them into production faster than ever. This is done by using rapid miner's lightning fast visual workflow designers and automated modeling capabilities. Rapid miner eliminates the complexities of cutting edge data science by making it easy to use in the latest machine learning algorithms and technologies like tensor flow, hadoop, and spark [22]. Rapid miner is based in Boston, Massachusetts, U.S. Its platform includes rapid miner studio, rapid miner server and rapid miner radoop. Rapid miner studio is a model development tool, available as both free and commercial editions; it is priced according to the number of logical processors and the amount of data used by a model [21]. Rapid miner provides learning schemes, models and algorithms and can be extended by using R and Python scripts [23]. In this article, the classification techniques of data mining used, are explained later.

A.1. Classification Technique

Classification is a data mining technique used to predict group membership for data instances within a given dataset. It is used for classifying data into different classes by considering some constrains. The problem of data classification has many applications in various fields of data mining. This is because the problem aims at learning the relationship between a set of feature variables and a target variable of interest. Classification is considered as an example of supervised learning as training data associated with class labels is given as an input [24]. In this article, different classification techniques that were used is explained in detail, below:

- **Decision tree:** Decision tree classification provides a rapid and useful solution for classifying instances in large datasets with a large number of variables. There are two common issues for the construction of decision trees: (a) the growth of the tree to enable it to accurately categorize the training dataset, and (b) the pruning stage, whereby superfluous nodes and branches are removed in order to improve classification accuracy [25].
- **K- Nearest neighborhood (K-NN):** In K-nearest neighbor (KNN) technique, the nearest neighbor is measured with respect to value of k that defines how many nearest neighbors needed to be examined in order to describe class of a sample data point. Nearest neighbor technique is divided into two categories i.e., structure-based KNN and structureless KNN. The structure-based technique deals with the basic structure of the data where the structure has less mechanism associated with training data samples. In structureless technique entire data is categorized into sample data point and training data, and the distance calculated between sample points and all training points and the point with smallest distance is known as the nearest neighbor [26]-[32].
- **Naïve Bayes:** Naive Bayes classifiers are simple probabilistic classifiers based on the Bayes theorem. These are highly scalable classifiers which involve a family of algorithms based on a common principle assuming that the value of a particular feature is independent of the value of any other feature, given the class variable. In practice, the independence assumption is often violated, but Naive Bayes classifiers still tend to perform very well under this unrealistic assumption [24].

To do so, COCOMO data set 81 in rapid miner software was used and 3 features were omitted due to being numerical and also being a unsupervised PCA method. Supervision of decreasing dimension in MATLAB software and related formulae were used and COCOMO data set 81 turned into a new data set. Therefore, as it was said before, new data sets were used to make and create models. In rapid miner software, the preprocess of data was done after choosing a new data. This phase includes choosing data sources, omitting diverted points, and how to treat the omitted data, and turning, extracting or decreasing. To decrease dimensions and extract the best features, the omitted purpose was added to the new data set in order to get the output from the new collection. To extract the purpose which is a real attempt, the related doer id is used and we also consider positive for high expenses and negative for low expenses. In order to create a model which aims to extract samples or hidden models, Gain-Ratio criteria and Euclidean distance are applied.

B. Data Collection

As it was said, we used 2 data sets of NASA 93 and COCOMO 81. Data set NASA 93 has the format of COCOMO 81 and includes 93 records and 24 features. COCOMO data set 81 includes 19 features and 93 records. Also, in the two data sets, 70% of data are used for teaching and 30% of data are used to test in rapid miner software. Features and amounts in both data set are shown in Tables 1 and 2.

TABLE 1
FEATURES AND THE AMOUNTS OF FEATURES IN NASA 93 DATA SET [5]-[6]

Attribute	Attribute Value
Project name	De, Erb, Gal, X, Hst, Slp, Y
Applied classification	Avionics, Application–ground, Avionics monitoring, Batch data processing, Operating system, Real data processing, Science, Simulation, Utility.
Ground or air system	F , G
Center of NASA	1, 2, 3, 4, 5, 6.
Development year	1971, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1982, 1983, 1984, 1985, 1986, 1987.
Development mode	Embedded, Organic, Semi-detached

In table 2, the amounts are XH, VH, H, N, VL which refer to very Low, low, nominal, high, very high, extra high.

TABLE 2
FEATURES AND THE AMOUNTS OF FEATURES RELATED TO COCOMO 81[5]-[6]

Attribute	Attribute Value
The ability of analysts (ACAP)	VL, L, N, H, VH, XH
Programmers ability (PCAP)	VL, L, N, H, VH, XH
Program experiments (AEXP)	VL, L, N, H, VH, XH
Modern planning practices (MODP)	VL, L, N, H, VH, XH
Use the software tool (TOOL)	VL, L, N, H, VH, XH
Virtual machine test (VEXP)	VL, L, N, H, VH, XH
Language testing (LEXP)	VL, L, N, H, VH, XH
Program limitation (SCED)	VL, L, N, H, VH, XH
Main memory limit (STOR)	VL, L, N, H, VH, XH
Database size (DATA)	VL, L, N, H, VH, XH
Time limit for CPU (TIME)	VL, L, N, H, VH, XH
Rotation time (TURN)	VL, L, N, H, VH, XH
Machine fluctuations (VIRT)	VL, L, N, H, VH, XH
The complexity of the process (CPLX)	VL, L, N, H, VH, XH
Software reliability required (RELY)	VL, L, N, H, VH, XH

C. Evaluation Criteria

From the literature, the evaluation metric can be categorized into three types, which are threshold, probability and ranking metrics. Each of these types of metrics evaluates the classifier with different aims. Furthermore, all of these types of metrics are scalar group method where the entire performance is presented by using a single score value. Thus, it makes easier to do the comparison and analysis, although it could mask subtle details of their behaviors. In practice, the threshold and ranking metrics are the most common metrics used by researchers to measure the performance of classifiers. In most cases, these types of metrics can be employed into three different evaluation applications [27]. Firstly, the evaluation metrics are used to evaluate the generalization ability of the trained classifier. In this case, the evaluation metrics are used to measure and summarize the quality of trained classifier when tested with an unseen data. Accuracy or error rate is one of the most common metric in practice used by many researchers to evaluate the generalization ability of classifiers. Through accuracy, the trained classifier is measured based on total correctness which refers to the total of instances that are correctly predicted by the trained classifier when tested with an unseen data. Secondly, the evaluation metrics are employed as an evaluator for model selection.

In this case, the task of evaluation metrics is to determine the best classifier among different types of trained classifiers which focuses on the best future performance (optimal model) when tested with an unseen data. Thirdly, the evaluation metrics are employed as a discriminator to discriminate and select the optimal solution (best solution) among all generated solutions during the classification training. For example, the accuracy metric is employed to discriminate every single solution and select the best solution that id produced by a particular classification algorithm. Only the best solution which is believed to be the optimal model will be tested with an unseen data [27]. Different features are as follows:

C.1. Accuracy Criterion

In general, the accuracy metric measures the ratio of correct predictions over the total number of instances evaluated [27]. The accuracy of classification is calculated according to the following function.

$$\text{Accuracy} = \frac{\text{TP + TN}}{\text{TP+FP+TN+FN}} \qquad (5)$$

where TP and TN denote the number of positive and negative instances that are correctly classified. Meanwhile, FP and FN denote the number of misclassified negative and positive instances, respectively [27].

C.2. Recall Criterion

Recall is used to measure the fraction of positive patterns that are correctly classified [27]. The following function shows how to calculate this criteria [28]-[29]-[30]-[31].

$$\text{Recall} = \frac{\text{TP}}{\text{TP+TN}} \qquad (6)$$

C.3. Precision criterion

Precision is used to measure the positive patterns that are correctly predicted from the total predicted patterns in a positive class [27]. This criteria is calculated by the following function [28]-[29]-[30]-[31].

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{7}$$

D. *Results*

In this part, we present the results of 2 data sets by using learning algorithm of machine and the suggested method of PCA and then compare these results. The results are shown in Tables and Charts below:

TABLE 3
THE RESULTS OF ASSESSMENT BY APPLYING 3 ALGORITHMS

Metric	Algorithm		
	Decision tree	Naïve Bayes	K-NN
Accuracy	60.71%	46.43%	53.57%
Precision	50.78%	47.16%	53.11%
Recall	50.64%	61.28%	52.82%

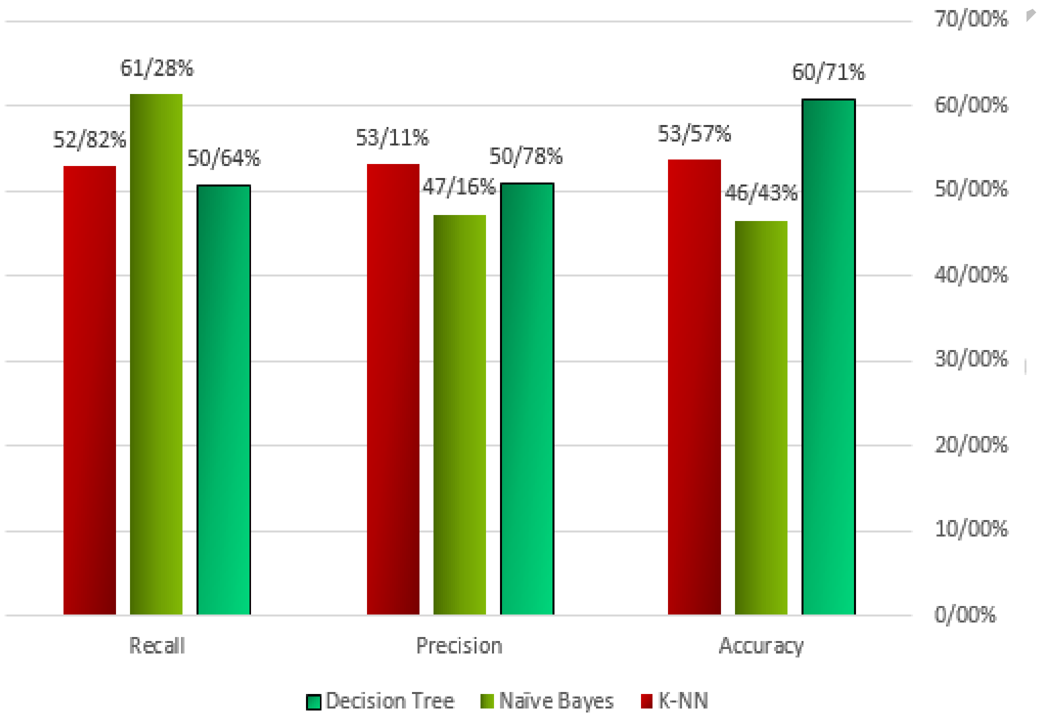


Chart 1: The results of assessment of the 3 algorithms.

The results related to the three algorithms by using PCA method, are shown below.

TABLE 4
THE RESULTS OF EVALUATION USING PCA

Metric	Algorithm		
	Decision tree	Naïve Bayes	K-NN
Accuracy	78.95%	84.21%	94.74%
Precision	68.33%	77.08%	96.88%
Recall	68.33%	71.67%	87.50%

The results show that tree algorithm in COCOMO NASA 93 with the accuracy of 60.71% is the best method.

But, since we used PCA in COCOMO data set 81, the authenticity algorithm with the accuracy of 94.74% was the best method.

E. *Analysis of the results*

As it was explained before, we analyzed our results in a way that the pre-process of data was used. To predict, classification techniques were used, but in our case, we used dimension decrease method.

In fact, we decreased the dimension by using the PCA method and turned it into a new data set.

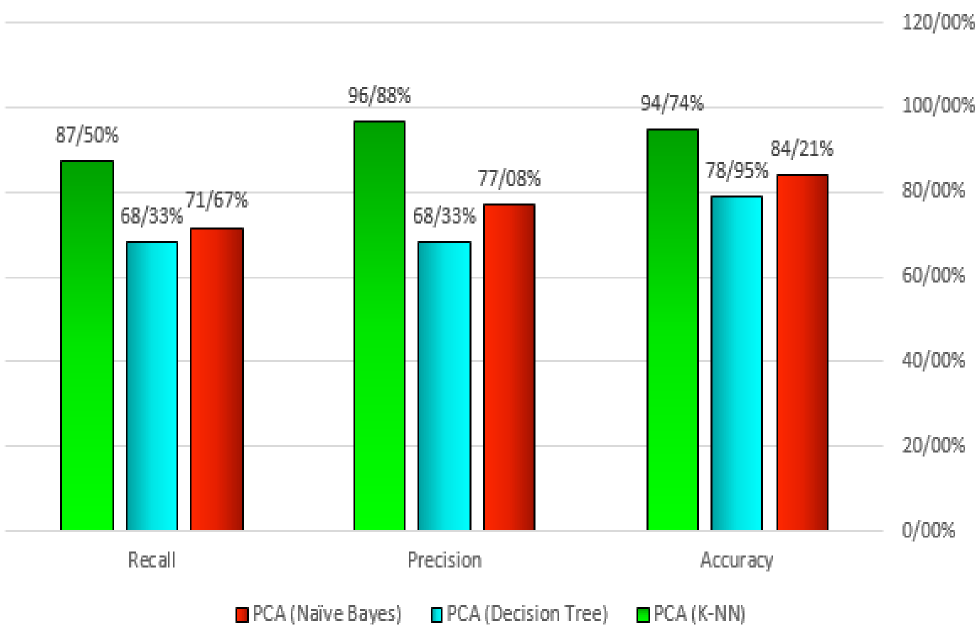


Chart 2: The results of evaluation of the 3 algorithms using PCA.

Then, we created predicted models by using learning algorithms of machines. The new data treatments were predicted and then validated by the models, but, we used data set COCOMO 81 due to its being numerical and being unsupervised version of the PCA method. Therefore, a search was done by using learning algorithms of machines and some samples were explored to predict new positions. We used different algorithms and the prediction was done based on the purpose features.

The prediction in these 3 algorithms are according to the purpose features (real effort): Naïve Bayes algorithm: the probability of software cost by increasing expenses (positive) in group 10 was 0.794 and probability of software cost estimation by decreasing expenses (negative) in group 10 was 0.206 and their authenticities were evaluated by the an accuracy of 84.21%.

The decision tree algorithm: based on decision tree model, the chance that these features decrease expenses are more, or which feature was F1, which showed the least cost (negative). The algorithm had branches in which positive and negative were put. The model shows that less expenses exist in branches (by using the existing features).

Therefore accuracy of predicted model was 78.95%. The K-Nearest Neighbor: the created model in the neighborhood (K = 1) were on all the samples with 10 dimensions in 2 groups of positive and negative and there accuracies were predicted to be 94.74%.

In this article, we presented the best method to increase accuracy in software cost estimation by using principal component analysis and learning algorithm of machine and decreasing costs.

6. Conclusion and Future Works

In this article, classification technique was used to estimate software cost. Therefore, principal components analysis method to decrease input data dimensions and classification algorithms to model and evaluate them on COCOMO data set 81 were used to increase accuracy.

The results of COCOMO 81 was compared with the results of NASA 93.

The results proved that the suggested method could have significant influence on models of decision tree, naïve Bayes and nearest neighborhood by decreasing dimension of input data and turning it into data. In this article, the most amount of accuracy is related to the most adjacent neighborhood algorithm with the accuracy of 94.74%.

In future, it is suggested to apply a different learning algorithm of machines and a different software work and also to use different methods such as wrapper in order to improve software cost estimations.

REFERENCES

[1] F. Soleimanian Gharehchopogh, A. Talebi, and I. Maleki, “Analysis of use case points models for software cost estimation,” *International journal of academic Research*, Part A, vol. 6, no. 3, pp. 118-124, 2014.

[2] H. Leung and Z. Fan, “Software cost estimation,” Handbook of Software Engineering, Hong Kong Polytechnic University, pp. 1-14, 2002.

[3] M. Fatima, S. F. Ahmad, and M. Hasan, “Fuzzy based software cost estimation methods: a comparative study,” *IJIRST-International Journal for Innovative Research in Science & Technology*, vol. 1, no. 7, pp. 287-290, 2014.

[4] R. Tripathi and P. K. Rai, “Comparative study of software cost estimation techniques,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 6, no. 1, pp. 323-328, 2016.

[5] T. Menzies, D. Port, Z. Chen, and J. Hihn, “Validation methods for calibrating software effort models,” presented at the 27th International Conference on Software Engineering, Saint Louis, USA, 2005.

[6] J. Hihn and T. Menzies, “Data mining methods and cost estimation models: Why is it so hard to infuse new ideas?,” in *Proc. 30th IEEE/ACM International Conference on* Automated Software Engineering Workshop (ASEW), pp. 5-9, Lincoln, USA, 2015.

[7] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn, “Negative results for software effort estimation,” *Empiriccal Software Engineering*, vol. 22, pp. 1-22, 2016.

[8] S. Gupta, S. Tiwari, H. Singh, A. Shukla, and H. Raghuvanshi, “A comparison between various software cost estimation models," *International Journal of Emerging Trends in Science and Technology*, vol. 3, no. 11 , pp. 4771-4776, 2016.

[9] T. Kaur and J. Singh, “A hybrid model for the enhancement in software effort estimation,” *International Journal of Scientific & Engineering Research*, vol. 6, no .7, pp. 619-624, 2015.

[10] S. Sharma, A. Kaushik, and A. Tomar, “Software cost estimation using hybrid algorithm,” *International Journal of Engineering Trends and Technology (IJETT)*, vol. 37, no. 2, pp. 62-71, 2016.

[11] A. khatibi Bardsiri and S. M. Hashemi, “Software effort estimation: a survey of well-known approaches,” *International Journal of Computer Science Engineering (IJCSE)*, vol. 3, no. 1, pp. 46-50, 2014.

[12] G. Mathew, T. Menzies, and J. Hihn, “Impacts of bad ESP (early size predication) on software effort estimation,” arxiv preprint arxiv: 1612.03240, pp.1-17, February. 2018.

[13] H. Najadat, I. Alsmadi, and Y. Shboul, “Predicting software projects cost estimation based on mining historical data,” *International Scholarly Research Network*, ISRN Software Engineering, vol. 2012, January 2012.

[14] I. M. Baytas, K. Lin, F. Wang, A. K. Jain, and J. Zhou, “Stochastic convex sparse principal component analysis,” *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 15, no. 1, pp. 2-11, 2016.

[15] T. Ensor, J. Cami, N. H. Bhatt, and A. Soddu, “A principal component analysis of the diffuse interstellar bands,” *The Astrophysical Journal*, vol. 836, no. 2, pp. 1-31, 2017.

[16] T. M. V. Suryanarayana and P. B. Mistry, Principal component regression for crop yield estimation, Springer, 2016.

[17] R. Tavoli, E. Kozegar, M. Shojafar, H. Soleimani, and Z. Pooranian, “Weighted PCA for improving document image retrieval system based on keyword spotting accuracy,” *in Proc. 36th International Conference on Telecommunications and Signal Processing (TSP),* pp. 773-777, Rome, Italy, 2013.

[18] R. Tavoli and F. Mahmoudi, “PCA-based relevance feedback in document image retrieval,” arXiv preprint arXiv: 1209.2274, 2012.

[19] M. Ghazanfari, S. Alizadeh, and B. Teimourpour, *Data Mining & Knowledge Discovery*, Third edition, Iran University of science and Technology, Tehran, 2008.

[20] J. Fan, Y. Liao, and H. Lin, “An overview on the estimation of large covariance and precision matrices,” *The Econometrics Journal*, vol. 19, no. 1, pp. 1-46, 2015.

[21] C. J. Idoine, E. Brethenoux, J. Hare, P. Krensky, N. Shen, S. Sicular, and S. Vashisth, (2018, February 22). Gartner magic quadrant for data science and machine learning platforms. Available: Http://www.rapid miner.com/ /resource/Gartner-magic-quadrant-data-science-platforms. Html.

[22] Boston, Mass, (2018, February 26). Rapid miner named a leader in the 2018 Gartner magic quadrant for data science and machine-learning platforms. Available: Http:// www.rapidminer.com/news-posts/rapidminer-named-leader-2018-gartner-magic-quadrant-data-science-machine-learning-platforms.html.

[23] D. Morris. (2013). Rapid miner – a potential game changer. Available:Http://www.en.wikipedia.org/wiki/rapidminer.html.

[24] K. Deshmukh, S. Raut, and J. Bhargaw, “An overview on implementation using hybrid naïve Bayes algorithm for text categorization,” *International Journal on Future Revolution in Computer Science & Communication Engineering*, vol. 4, no. 3, pp. 142-146, 2018.

[25] D. M. Farid, L. Zhang, C. M. Rahman, M. A. Hossain, and R. Strachan, “Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks,” Expert System with Applications, vol. 4, no. 4, pp. 1937-1946, 2014.

[26] A. A. Soofi and A. Awan, “Classification techniques in machine learning: applications and issues,” *Journal of Basic & Applied Sciences,* vol. 13, pp. 459-465, 2017.

[27] M. Hossin and M. N. Sulaiman, “A review on evaluation metrics for data classification evaluation,” *International Journal of Data Mining Knowledge Management Process (IJDKP),* vol. 5, no. 2, pp. 1-11, 2015.

[28] M. Keyvanpour and R. Tavoli, “Document image retrieval: Algorithms, analysis and promising directions,” *International Journal of Software Engineering and Its Applications,* vol. 7, no. 1, pp. 93-106, 2013.

[29] R. Tavoli, “Classification and evaluation of document image retrieval system," *Wseas Transactions on Computers*, vol. 11, no. 10, pp. 329-338, 2012.

[30] M. Keyvanpour, R. Tavoli, and S. Mozafari, “Document image retrieval based on keyword spotting using relevance feedback,” *International Journal of Engineering*, *IJE Transactions A: Basics*, vol. 27, no. 1, pp. 7-14, 2014.

[31] M. Keyvanpour and R. Tavoli, “Feature weighting for improving document image retrieval system performance,” arXiv preprint arXiv: 1206.1291, 2012.

[32] M. Hasanluo, F. Soleimanian Gharehchopogh, "Software cost estimation by a new hybrid model of particle swarm optimization and k – nearest neighbor algorithms," *Journal of Electrical and Computer Engineering Innovations JECEI,* Vol. 4, No. 1, pp. 49-55, 2016.

BIOGRAPHIES

Azin Saberi Nejad received the Associate degree in 2013, the B.Sc. degree in 2015, and the M.Sc. degree in 2017, all in computer engineering software from Pooyandegan Danesh Institution of higher Education, Chalus, Iran. Her research interests is data mining.

Reza Tavoli is an assistant professor of department of computer engineering, Islamic Azad University of chalus, Chalus, Iran. He received his B.Sc. (2007) in software engineering from Iran University of Science & Technology, Behshahr, Iran. He received his M.Sc. (2009) in software engineering from Islamic Azad University, Science & Research Branch, Tehran, Iran. In addition, he received his Ph.D. (2018) of Software engineering from Islamic Azad University, Qazvin Branch, Qazvin, Iran. His research interests Include document image retrieval and data mining.

How to cite this paper:
A. Saberi Nejad and R. Tavoli “A method for estimating the cost of software using principle components analysis and data mining,” *Journal of Electrical and Computer Engineering Innovations*, vol. 6, no. 1, pp. 35-44, 2018.
DOI: 10.22061/JECEI.2018.811
URL: http://jecei.sru.ac.ir/article_811.html