Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 11, No. 3, 2023, 453-476.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Accuracy Improvement in Software Cost Estimation based on Selection of Relevant Features of Homogeneous Clusters

Saba Beiranvand[1*] and Mohammad Ali Zare-Chahooki[2]

1. Department of Computer Engineering, Technical and Vocational University (TVU), Tehran, Iran.
2. Department of Computer Engineering, Faculty of Engineering, Yazd University, Yazd, Iran.

Article Info

Article History:
Received 01 March 2023
Revised 01 May 2023
Accepted 01 August 2023

DOI:10.22044/jadm.2023.12750.2429

Keywords:
Software Cost Estimation (SCE), Software Effort Estimation (SEE), Machine Learning Methods, Clustering, and Feature Selection.

**Corresponding author: bir.saba@yahoo.com (S. Beiranvand).*

Abstract

Software Cost Estimation (SCE) is one of the most widely used and effective activities in project management. In machine learning methods, some features have adverse effects on accuracy. Thus, pre-processing methods based on reducing non-effective features can improve accuracy in these methods. In clustering techniques, samples are categorized into different clusters according to their semantic similarity. Accordingly, in the proposed study, to improve SCE accuracy, first samples are clustered based on original features. Then a feature selection (FS) technique is separately done for each cluster. The proposed FS method is based on a combination of filter and wrapper FS methods. The proposed method uses both filter and wrapper advantages in selecting effective features of each cluster, with less computational complexity and more accuracy. Furthermore, as the assessment criteria have significant impacts on wrapper methods, a fused criterion has also been used. The proposed method was applied to the Desharnais, COCOMO81, COCONASA93, Kemerer, and Albrecht datasets, and the obtained Mean Magnitude of Relative Error (MMRE) for these datasets were 0.2173, 0.6489, 0.3129, 0.4898, and 0.4245, respectively. These results were compared with previous studies and showed improvement in the error rate of SCE.

1. Introduction

Software cost estimation (SCE) is an important phase of the software project development process. In this phase, the required cost for software development and maintenance is predicted based on various projects, and their features are usually incomplete, uncertain, and noisy [1]. In more research works; the SCE term is often used equivalently as the software development effort estimation (SDEE). If at the beginning of the project, the costs are not estimated in the SCE phase, precisely, the project may fail in the middle of its process. Therefore, since 1980, various methods have been proposed to estimate the software costs. These SCE methods can be divided into three categories; (1) algorithmic models, (2) expert judgment techniques, and (3) machine learning methods. SCE in algorithmic methods is done through (1).

$$EFFORT = F(x_1, x_2, x_3, \ldots, x_n). \quad (1)$$

In Equation (1), the variables x_1, x_2, x_3, …, x_n denote the features of each software project. In the algorithmic approach, a model is formulated based on a specified algorithm, and the obtained equation is used for estimating software costs. So far, many different algorithmic models have been proposed for SCE. The constructive cost model (COCOMO) is the most well-known method in this category [2]. This model is used for predicting the development time and staff size for a software development activity per month. Furthermore, it estimates the effort for each phase of a software development activity. Since algorithmic models are based on old

data, they cannot reflect current advances in programming languages, hardware, and software engineering with accurate software cost estimation [3].

The expert judgment technique is the second category of SCE methods. Old data is usually not needed in this method. Experts' judgments are often based on the estimators of previous projects which may have not been documented. Studies show that 62% of estimators use this method in their organizations [3]. The advantage of these expert-based methods is that they have been customized for a certain organizational culture, and as proved in so many cases, they are more precise than other models. However, this kind of estimation is entirely subjective, and is built upon personal logic. Thus, its advantage can also be considered as its disadvantage. This is because experts would estimate the costs based on (1) their specific experiences and (2) a certain organizational culture [3]. Therefore, the same expert may do the estimation less precisely in another organization.

The third category of SCE is the approach of using machine learning methods. These methods learn patterns from obtained data of previous projects, and use them for cost prediction. The basic idea before using machine learning methods for effort estimation is that the old data include many old projects that have been described through their valuable features and projects with similar characteristics also include almost similar project efforts. So far, many specialists have used different machine-learning techniques in this regard and achieved promising results [2, 3, 4, 5, 6] .

Therefore, using a machine learning approach along with the data of previous developed projects, a model can be achieved to estimate the required cost for new projects. Due to the uncertainty of software cost estimation, using uncertain and flexible machine learning methods can play an important role in enhancing the performance of estimations. The advantage of such estimation methods is ability to do the complex relationship between project cost and features by learning data of previous projects [4]. Finding an efficient subset of features that is used for improving the accuracy of a learning model is challenging. This is due to the vulnerability of machine learning techniques to erroneous, irrelevant, and redundant features [6]. SCE models use a large set of features called cost determinants. However, not all of these features are effective for an accurate estimation. Thus, feature subset selection algorithms are used in the SCE phase to improve the accuracy of models by selecting the most informative cost determinants [5].

Feature selection methods are classified into three embedded, filter, and wrapper groups. In embedded methods, the feature selection function is integrated with machine learning techniques. The C4.5 decision tree is the most famous embedded feature selection algorithm. Filter methods perform a feature selection process using simple measures, but their results are not highly accurate. Wrapper methods select the optimal subset of features on the iterations of training and validation of a given learning method. Although these methods are better than filter methods in terms of finding useful features, they are so slow, specifically having a large set of features [7]. Hence, this study aims to offer a new method using the most effective features of software projects that it can be selected through machine learning techniques to have a more precise estimation by implementing a hybrid filter-wrapper method for feature selection.

Existing datasets in the field of software engineering and software cost estimation suffer from noise and outliers [8]. Machine learning algorithms create their model from training data, so noise and outliers reduce the accuracy of these methods.

Clustering methods classify homogenous samples in similar clusters. On the other hand, it is expected that the feature selection of homogenous samples is more efficient. Therefore, the final feature selection in this paper is based on homogenous samples using clustering and a combinatorial evaluation criterion. Main represented innovation in our paper propose the effective feature selection for samples in homogeneous clusters of software projects. Since cost estimation in software projects is closely related to the nature of the project, adding this assumption to the process of feature selection can highly enhance the efficiency of software cost estimation. Wrapper methods in feature selection have the limitation of time complexity. So, we have used a hierarchical structure of filter and wrapper methods. Various combinations of filter methods have been evaluated in this approach. The most effective filter feature selection method has been combined with one of the most effective wrapper methods in cost estimation for software projects. Evaluation criterion in wrapper techniques has a direct impact on effective feature selection. Therefore, a combinatorial criterion is used for SCE in wrapper feature selection. The HH-FS method was tested in our previous study in the field of software cost estimation [9], and the results show that the idea is effective on increasing

accuracy. Therefore, in this article, we test the same method on homogeneous data obtained from clustering.

In the following of this paper, in the second section, related works will be reviewed. In the third section, the proposed method will be introduced. The experimental results of implementing the proposed method in the field of software cost estimation along with the analysis process will be discussed in the fourth section. In the end, the conclusion and suggestions for future work are presented in the fifth section.

2. Related Works

Software development is the process of designing, creating, testing, and maintaining different software. It involves the use of various principles and techniques from computer science, engineering, and mathematical analysis. The aim of software development is to create efficient, reliable, and easy-to-use software. There is a variety of software development methodologies that can be used to create software applications. The most popular methods include waterfall model, agile model, and spiral model.

There are two main methods of software development: predictive and adaptive.

- In forecasting method, the requirements and schedule are known in advance and the project is planned and executed accordingly. And the project was implemented in the waterfall methodology.
- In adaptive method, the requirements and schedule are not known in advance, and the project is executed in an agile and iterative manner.

Thus, which method should you choose? It depends on the project you are working on.

SDEE is the process of predicting the required effort for the software development life cycle. Researchers [10] review the most recent machine learning (ML) techniques used to SCE for both, non-agile and agile methodologies. [11] presents a study to identify cost/effort methods in Agile Software Development (ASD), standards, and issues, are carried out through systematic mapping. It included studies that show how the estimation of cost in ASD has evolved on both the number of used methodologies and the new implementation of machine learning techniques, to present new models to estimate the applied effort and automate the estimation process, accurately. Temporal data mining is a field that is developing rapidly, this work provided a detailed overview of various methods for mining temporal sequences [12]. SCE is necessary for all software organizations to contract negotiation. Researchers have worked over the last four decades to provide estimation models, which can estimate efforts as accurately as they could, but the continuous change in the development models and use of new programming languages requires the development of new techniques for cost estimation [4].

The product size and the cost drivers organize effort predictors [13]. SDEE not only could be considered a development effort, but also it could include the required effort for the maintenance process. As mentioned earlier, accurate estimation of required effort in the early process of software development plays an important role in project management [14]. At the beginning of process, software project managers need reliable methods for performing works such as possibility studies, planning, resource allocation, and cost or effort estimation, at the initial stages of software development [15]. Since 1980, when the most fundamental changes in the field of software effort estimation have occurred, expert judgment-based and machine-learning methods have been used more than other methods [16]. In [17], researchers provide a survey of SCE models and summarize their strengths, weaknesses, accuracy, amount of needed data, and validation used techniques.

Artificial intelligence algorithms can improve the efficiency of used methods in estimations by searching for the appropriate configuration for formulated SCE methods [18]. Since 1991, eight major groups of machine learning algorithms have been used in SCE [16]. These methods include case-based reasoning (CBR) [6], artificial neural networks (ANN) [4], decision trees (DT) [17], Bayesian networks (BN) [19], support vector regression (SVR) [4], genetic algorithms (GA) [5], genetic programming (GP) [20], and association rules (AR) [21]. GA is used only in synthetic form and combination. It is used only for weighting and feature selection. Fuzzy logic is used to overcome the uncertainty and inaccuracy of information. In practice, fuzzy logic is used for improving model performance by pre-processing the model inputs [16]. In addition to the proposed categorization, the important point is that new machine-learning methods have been proposed in recent years and used in this area. In [22], the proposed model uses spiking neural networks to improve the accuracy of estimated process cost which is improving the quality of the software.

To improve the performance of machine learning methods, training datasets require pre-processing in most areas of study including SCE. Normalization [19, 20] and feature selection are commonly used approaches in SCE. In [17], PCA

has been used for dimension reduction as preprocessing and has shown that this preprocessing led to more accuracy. Numerous research studies have used machine learning and feature selection methods for software cost estimation. Studies including [23] have used Feature Subset Selection (FSS) for increasing the accuracy in SCE and offering the nearest estimated value to actual effort for the available samples. Researchers in [22] have stated that FSS is among the necessary tasks in SCE. Hosni *et al*. [24] compare the fuzzy analogy ensembles built without using feature selection with ensemble fuzzy analogy and filter single techniques.

In [5], researchers have assessed the association between features of historical datasets to reduce cost determinants with maintaining performance. In their study, nine FSS popular methods have been used for selecting the most effective cost determinants and the results indicated that FSS is effective to achieve optimal accuracy in SCE. In [25], a combined method based on mutual information and feature clustering is provided. It can be said that in their study, unsupervised and supervised learning methods are combined. In the unsupervised learning phase, features are classified in several clusters based on the similarity between features and the obtained clusters by using hierarchical clustering. Then in the supervised learning phase from each cluster, the feature that has the most similarity to the effort feature is selected as the representative of that cluster. The results indicated the effectiveness of this combined method in optimizing the accuracy of the SCE.

In [26], various multiple methods were investigated, and it was shown that collaborative methods provide the best accuracy. Accordingly, various learning-based models in combination with other algorithms have been proposed. Elish *et al*. in [4] have proposed different homogeneous and heterogeneous combinations of different machine learning algorithms and have done various experiments on them. Mary *et al*. [27] improve the accuracy and sensitivity of COCOMO-II model with the combination of Artificial Neural Networks and Fuzzy Logic (Neuro-Fuzzy models). In [28], a random forest (RF) model is used. They have optimized their model empirically by changing the values of its key parameters. The accuracy of the RF and regression tree is compared.

Each FSS algorithm may be examined from both efficiency and effectiveness perspectives. Efficiency is related to the required time for finding the optimal subset of original features and effectiveness is related to the quality of the selected subset. According to these perspectives, in [29], a PSO tuning algorithm is used for data clustered by the K-Means clustering method. The accuracy of the used model for predicting the effort that is required for software project development is dependent on data parameters. In [30], the classification accuracy in software defect prediction is evaluated for data with and without noise deletion. The results showed that data cleaning improved the accuracy.

In [6], an extensive search has been done to find the best subset of cost determinants with hill-climbing and forward wrapper methods. In [27], linear regression and FSS based on wrapper methods are used and the authors concluded that effort estimation has been improved by selecting effective features.

In [28], linear regression with selecting the most effective features has been implemented based on FSS wrapper-based methods. Results indicated that pruning rows and columns (samples and features) can significantly improve the results of effort estimation, especially in small datasets. In their research, by considering the results of experiments, the authors concluded that pruning rows are effective in reducing the mean and standard deviation in evaluated models.

In [31], project features were identified by the COCOMO model, and then using FSS wrapper-based technique, effective features were selected. Although based on previous research FSS wrapper-based method was mentioned as a good method. In [28], it is indicated that accurate results by using this method are not efficient due to the high computing cost and the impossibility of generalizing selected subset of features for use in other models. In [32], feature weighting and also comparative methods based on euclidean distance led to an accurate estimation of the required effort for the production and development of software projects.

In [33], the accuracy of ordinary least squares (OLS) has been evaluated for clustered data and data without clustering. For clustering, two forms of feature selection have been used. First, features with scale ratio were selected by the Pearson correlation method, and nominal features were selected by the ANOVA method. Then samples are clustered based on the selected features of Pearson by K-means and the selected features of ANOVA by Scheffe's method and then the two clusters that have similar samples in these two models are merged which it results in the reduction of selected features. The evaluation results of OLS for SCE of the ISBSG dataset indicated that the accuracy of the two proposed methods was not so much different. In [34], the fuzzy clustering method has

been used to dominate the heterogeneous nature of samples of the ISBSG dataset.

Software project managers have two approaches for error reduction of required effort for software development: 1) changes based on general experience (data collected from production projects of different companies) and 2) modifications based on personal experiences (data of projects are generated locally in company. In [31], it is shown that the defect prediction of effort estimation performance of the model based on data that is similar to the test samples is better than the performance of models that is developed based on non-relevant data. Since Kashin's learning algorithms build their model based on training data, the irregularity and presence of noise in the data used to train machine learning algorithms can hurt the accuracy of these algorithms. In [35], a comprehensive and systematic assessment has been done on combinatorial models such as bagging in combination with multi-layer perceptron (MLP) neural network and local approaches such as clustering methods that use similar samples for learning to estimate the cost of new samples.

In [36], an ensemble method is proposed for classification based on feature extraction. To create training data for classification, features were divided into k categories, and then the PCA feature extraction method was applied to each of the categories. The main idea is the data rotation and displacement of feature vectors and samples. Since the Decision Tree (DT) is used for classification, and this classification method is sensitive to feature vectors and displacement, this method is called rotation forest. To increase the accuracy and decrease the time complexity, Madari *et al*. [37] proposed a hybrid model of FSS and regressor named RF, XGBOOST, and K-Means with hamming distance and multi-layer perceptron.

It is indicated that features of software projects are very complex, non-linear, and incomprehensible due to the uncertain and unstable nature of software projects [38]. Non-algorithmic methods use one dataset for estimation that it includes a variety of software projects. Such datasets include many unrelated and contradictory projects which are considered outliers in data mining. In machine learning methods, these outliers could adversely affect the quality of training, and may lead to inaccurate and unreliable estimations. This issue has been addressed and resolved by clustering the projects. The researcher’s suggestion in [39] to overcome the problem of outliers is neural network training and analogy-based estimation (ABE) based on obtained clusters.

Numerous SCE strategies for software development applying classical methodologies may be discovered in the papers. However, these strategies are not well-suited for other development methodologies, because variety of sizes is necessary for estimating [40]. The Case-Based Reasoning (CBR) methods are among those methods that rely on the history of successful previous projects to predict solutions for new projects [41]. Although in the recent years, machine learning-based models have been used more for the estimation of software development, none of the existing models are suitable for all project conditions and their performance varies for different datasets. Therefore, it is essential to develop a highly reliable model for new projects [4].

3. Proposed Method

A selected subset by an evaluation function may not be the same as a selected subset by another, because a subset is better 2 that depend on the used methods for this work [42]. Despite the opportunities have been created by multi-dimensional datasets, such datasets cause a lot of computational challenges. One of the problems with multidimensional datasets is that in most cases all of their features are not vital to find the knowledge that is hidden inside them [43]. In SCE, machine learning methods extract a model using the relationships between features describe training samples and then use this model to estimate the software cost of a new sample. So, undoubtedly, features describe samples have a significant impact on the accuracy of machine learning methods. In other words, the accuracy and correctness of the model highly depend on sample features [43].

Software cost estimation using the full list of available data features may lead to a reduction in the effort estimation accuracy [31]. Some of these features may be irrelevant or redundant and cause additional noise and complexities. Because of that, data dimensionality reduction and especially feature selection has remained a considerable discussion. Feature selection methods attempt to select a subset of basic features to reduce data dimensions. This has great importance in many applications because there are a large number of features in many situations which are redundant and irrelevant. Numerous solutions and algorithms have been proposed to solve the issue of feature selection, some of them back thirty or forty years ago [44]. What we obtain from feature selection is a subset reduced from the set of original features. The aim of feature selection is normally to identify the most important and influential features of the

dataset to achieve optimal performance in terms of speed, prediction, simplification of the model, data virtualization for model selection, dimension reduction, and noise removal [44]. The rest of the features are known as redundant or irrelevant features [44].

Feature selection methods include two major filter and wrapper categories. Filter methods work so rapidly using a simple measurement criterion but the results are not satisfactory in most cases. On the other hand, wrapper methods ensure acceptable results through the evaluation of learning outcomes but they are slow. Thus, working with high-dimensional data, wrapper techniques are not suitable in terms of computational speed. Even though filter methods are proper for dealing with such data, but they are unstable in terms of accuracy [44]. This paper aimed to use a combined technique consisting of both principal feature selection methods to choose the most effective features. This combination is slower than filter methods but it is more accurate. So far, various combined feature selection methods have been proposed in different fields [44], but the presented method in this study differs from the other ones. In this study, it has been tried to select the relevant features by considering the dispersion of the data samples with a combination of the most widely used feature selection methods and by the trade-off between the evaluation criteria.

Considering the advantages and disadvantages of two widely used feature selection methods (wrapper and filter), we can combine them to use the benefits of each one to fix the defects of the other one. Filtering methods are based on statistical criteria in which the accuracy measure is not taken into consideration. Therefore, since they do not need to run learning methods and examine their accuracy level, they work fast. On the other hand, because of this advantage, the process will be less accurate when they are used along with machine learning techniques. Besides wrapper techniques run the learning algorithms over the process of feature selection and evaluate their accuracy, it is obvious that they will reach an appropriate level of precision. However, they have to evaluate all feature subsets. Thus, they become slow. Hence, the number of steps that required to assess the features in wrapper methods can be reduced by implementing filter methods in the first step. Since wrapper methods are used in the next step to examine the selected features in the previous step, the accuracy of learning methods will also be increased.

Machine learning, as the name suggests, involves learning systems from existing data using algorithms that iteratively learn from datasets and analyze the data to develop or train model. Machine learning methods make a training model using available data in the dataset upon which they perform the estimation operation. It is clear that outlier data affects the training quality of these algorithms thus estimates will be inaccurate and unreliable. Since the available data in software cost estimation have the problem of outliers and noise [8], hence, to solve this problem, clustering of the projects is used in this study. To overcome the contradiction and diversity of information among a variety of samples, a learning process is required to be done in a set of homogeneous projects. Clustering can increase project consistency; thus, learning accuracy could be improved by putting similar projects in similar clusters. Instead of having a set containing multiple heterogeneous samples, several subsets with consistent and similar samples can be achieved. Thus, in the present study, we first cluster the samples to decrease the effect of heterogeneous samples on the quality of the learning algorithm. Similar samples in the same clusters may have different features from samples in other clusters. Thus, by this hypothesis, feature selection is separately applied to each cluster for accuracy improvement of software cost estimation. Accordingly, selecting the most effecting features for the homogenized data through clustering techniques is the novelty of this approach. In this regard, a new combined method of Hierarchical Homogeneous Feature Selection (HH-FS) is offered for feature selection.

K-means, based on various researches is a widely used algorithm for homogenization of samples [45, 46, 47]. The K-Means algorithm is an iterative-based algorithm that tries to define the data set into distinct subgroups without overlapping, which are called clusters; in these groups, each data point belongs to only one group. In this algorithm, we try to make the data points in a cluster as similar as possible, and at the same time define the clusters as different (far apart) as possible. The k-means clustering method is the simplest method for data clustering. Among the advantages of the K-Means algorithm, the following can be mentioned:

- Very high speed of this method in execution.
- The use of this algorithm is very simple (using ready-made libraries and packages related to this algorithm).
- This algorithm can be used for large amount of data.
- This algorithm guarantees convergence.
- This algorithm considers the best positions for centers.

- This method is easily adapted to new samples.
- Clusters with different shapes and sizes are generalized like elliptical clusters.

Therefore, among the different methods, the K-Means method was chosen for clustering.

The proposed method in this paper is expressed in two main sections: To overcome the contradiction of samples, training data are clustered by using K-Means. K-Means clustering in MATLAB is used, and initial points are selected without user intervention. In the next step, for each cluster, effective features are selected for accuracy improvement by combining filtered and wrapper methods. To select the effective features, at first, P filter selection methods are implemented on the dataset without clustering, each ranking features based on a specific statistical measure, and then an ordered list of features is available as an outcome. In this list, features with higher ranks are more important. The best features are selected from the output of every TP method. So far, there are P feature categories with TP features for each method.

It is clear that some of the features are common to all P feature categories, and other features are non-common. To separate these two categories of features, the "AND" and "XOR" operators are used. The output of the "AND" operator is a common feature. These common features are important because they are highly ranked by all P filtering methods with different criteria.

To determine the best features, two the simplest and widely used methods of wrapper methods named backward feature selection (BFS) and forward feature selection (FFS) [44], are used. These methods determine the effectiveness of each feature by assessing the accuracy of the learning algorithm. In this paper, the fusion function is used to combine the main measures of accuracy evaluation. This function uses the measures as inputs and proposes the new measure as the output function. The output is a combined measure that is used for accuracy evaluation. The advantage of using the fusion function in feature selection is evaluation of the impact of features, based on several criteria, simultaneously. This manner as shown in experimental results improves the accuracy of SCE. Common features are considered as primary features of samples and BFS tries to remove those features so that their deletion leads to accuracy improvement. This continues until the removal of any feature does not increase accuracy. FFS is used for selecting those features among non-common features that adding them to the selected feature set increases the accuracy in SCE. This continues until adding any non-common feature to the selected features set does not increase accuracy. If in this stage, the selected subset changes, all feature selection actions will be repeated with the newly selected subset. These changes mean that the estimation accuracy is increasing. At the end of this step, besides selecting the most effective features in data with one cluster, the best combinations of filter methods are selected for each dataset.

In the next step, the dataset is clustered and the best combination of feature selection methods is implemented on each cluster so that the most effective features are selected for every cluster. To estimate the software cost of a new sample, at first, the cluster of the sample is determined based on its similarity to the center of each cluster, and the effort is estimated using the regression algorithm for the new sample by considering the selected features of that cluster.

The proposed method in this paper can be expressed in the form of three main algorithms. Model learning is described in Algorithm 1. The output of Algorithm 1 is the training model for each cluster. In this algorithm, by calling Algorithm 3, which has the responsibility for selecting relevant features for each cluster, the most effective features are selected. The proposed method for HH-FS training consists of two main parts. In the first part, dataset clustering is done to resolve the issue of diversity and contradiction between samples. Then in the second part, combined feature selection is done to specify the effective features in each cluster. The most effective features of each cluster are specified in Algorithm 3. The condition "result is improved" in Algorithm 1, considers the situation that the selection process continues until the accuracy does not improve anymore. In Algorithm 3, first, the features of each cluster are ranked based on P filter feature selection methods, and the TP number of features with the best ranking in each filter procedure is chosen as selected features of that specific method. Algorithm 2 represents the process of cost estimation for the new samples.

The same filter-wrapper hybrid algorithm is used in [44]. The only difference is that the proposed method in [44] is applied for classification and this is the first time that a combination like this is used for software cost estimation on a regression model with homogeneous samples. In this study, also a combined evaluation measure is created using the fusion function. We previously tested the hierarchical features selection (HFS) with a combined criterion in [9] on the three original datasets (Desharnais, Cocomo81, and Coconasa93), and the results indicated the

effectiveness of the proposed method in the field of software cost estimation.

The system architecture of the proposed Hierarchical feature selection (HFS) is shown in Figure 1 and Figure 2 depicts the main steps of the proposed Hierarchical Homogeneous Feature Selection (HH-FS) method.

The system architecture of the proposed Hierarchical feature selection (H-FS) is shown in Figure 1 and Figure 2 depicts the main steps of the proposed Hierarchical Homogeneous Feature Selection (HH-FS) method.

ALGORITHM 1. MODEL LEARNING IN HH-FS (X, M, F)

Input:

$X = \{x^t, r^t\}_{t=1}^{N}$ // *where* x^t *is an instance,* r^t *is its associated label and N is the number of training instances. Also, any* x^t *is represented as* $[x_1, x_2 \ldots x_D]$ *where D is the number of features.*

$M = \{m_i\}_{i=1}^{K}$ // *where* m_i *is i*th *measure criterion in the application and K is the number of measurement criteria.*

$F = \{f_i\}_{i=1}^{P}$ //*where* f_i *is i*th *filter method and P is the number of filtering methods*.

Process:

T = Train set of X
V = Validate set of X where X=T∪V
c =1
do
$\{T_c, h_c\}_{c=1}^{C}$=*K-Means(T,c)*
$\{V_c\}_{c=1}^{C}$=*Assigning validation samples to each cluster where* V_c *is validation samples of* h_c

for *c=1: C*
$\boldsymbol{s_c}$=*Hierarchical FS algorithm(*$\boldsymbol{T_c}$ *, M, F)*
$Model_c$=*Model Training(*$\boldsymbol{T_c}$*,* $\boldsymbol{s_c}$*)*
$\boldsymbol{m_c}$ =*Model Testing(*$\boldsymbol{V_c}$*,* $\boldsymbol{Model_c}$*,* $\boldsymbol{s_c}$*)*
result=result+ $\boldsymbol{m_c}$
end
while (result is improved)

Output:

$H=\{\boldsymbol{T_c}, \boldsymbol{h_c}, \boldsymbol{s_c}\}_{c=1}^{C}$ // *where* $\boldsymbol{T_c}$ *is instances in the kth cluster,* $\boldsymbol{h_c}$ *is the center of the kth cluster and* $\boldsymbol{s_c}$ *is the optimum subset of original features in the kth cluster.*

ALGORITHM 2. ESTIMATION WITH HH-FS (H,x)

Input:

$x=\{x^t, r^t\}_{t=1}^{N}$ // *where* x^t *is an instance,* r^t *is its associated label and N is the number of instances. Also any* x^t *is represented as* $[x_1, x_2 \ldots x_D]$ *where D is the number of instance features. X is divided into the Training set and the Testing set.*

$H=\{T_c, h_c, s_c\}_{c=1}^{C}$ // *where* T_c *is instances in the kth cluster,* h_c *is the center of the kth cluster and* $\boldsymbol{s_c}$ *is the optimum subset of original features in the kth cluster.Process:*

$TE=\{x^t, r^t\}_{t=1}^{N}$, *Testing set of X where X=Training set ∪ Testing set*
for *j=1: N*
Max=0
for i=1:c
$C\_Similarity(j,i)=similarity(TE_j, h_c)$
If (max<C_ Similarity(j,i))
Max= C_Similary(j,i)
cluster=i
End if
end
$Model_J$=*Model Training(*$\boldsymbol{T_{claster}}$*,* $\boldsymbol{S_{cluster}}$*)*
$\boldsymbol{r_j}$ =*Model Testing(*$\boldsymbol{TE_j}$*,* $\boldsymbol{Model_j}$*,* $\boldsymbol{s_{cluster}}$*)*
end

Output*:*

r // where r is the estimation of x by the learned model.

ALGORITHM 3. HIERARCHICAL FS ALGORITHM

Input:

$X=\{x^t, r^t\}_{t=1}^{N}$ // *where* x^t *is an instance,* r^t *is its associated label and N is the number of training instances. Also any* x^t *is represented as* $[x_1, x_2 \ldots x_D]$ *where D is the number of instance features.*

$M= \{m_i\}_{i=1}^{K}$ // *where* m_i *is ith measure criterion in the application and K is the number of measurement criteria.*

$F= \{f_i\}_{i=1}^{P}$ //*where* f_i *is ith filter method and P is the number of filtering methods*.

Process:

for p=1: P

S_p = filter (f_i, *X*, t_p) *where Sp is a sorted set of top* t_p *are the selected features by* f_i *on X.*

end

m=fusion (M) // where fusion returns a fused measurement criterion

$A=AND_1^P$ *Sp,* $B=XOR_1^P$ *Sp, s= A*

mf=Regression(X, s,m) //where mf is the accuracy result evaluated by m

repeat

[s A mf]= BFS-Function (X,s, A, m, mf), [s B mf]= FFS-Function (X,s, B, m, mf)

until (*mf* is not better than previous values)

Output:

s *// where s is the optimum subset of original features,* ***m*** *//where m is the fused measurement criterion*

ALGORITHM 4. BFS-FUNCTION (X,S, A, M, MF)

Input:

$X=\{x^t, r^t\}_{t=1}^{N}$ //*where* x^t *is a sample,* r^t *is its associated effort and N is the number of samples. Also any* x^t *is represented as* $[x_1, x_2 \ldots x_D]$ *where D is the number of sample features.*

s //*where s is the initial subset for backward FS.*

A //*where A is an additive subset for backward FS*.

m //*where m is the measurement criterion*.

mf //*where mf is the accuracy result of the previous step*.

PROCESS:

n=1,Max=Size(B)

while (n ≤ Max)

while(f=selected next element of A)

S=s−f

[accuracy] = regression(X, S, m)

If accuracy>Best result in this iteration

Best=accuracy

b=f

end

end

if Best> mf

s=s – b, A=A – b, mf=Best

else

break

end

n=n+1

end

Output:

s *//where* ***s*** *is the optimum subset of features,* ***Mf*** */where mf is the accuracy of regression form S features.*

ALGORITHM 5. FFS-Function (X, s, B, m, mf)

Input:

X=$\{x^t, r^t\}_{t=1}^{N}$ *//where* x^t *is a sample,* r^t *is its associated effort and N is the number of samples. Also any* x^t *is represented as [*x_1, x_2… x_D*] where D is the number of sample features.*

s //*where s is an initial subset for forward FS.*

B //*where B is an additive subset for forward FS.*

m //*where m is the measurement criterion.*

mf //*where mf is the accuracy result of the previous step.*

PROCESS: FFS-Function (X, s, B, m, mf)

n=1; Max=Size(B)

while (n ≤ Max)
while (f=selected next element of B)
S=s ∪ f

[accuracy] = Regression (X, S)

If accuracy>Best result in this iteration
Best=accuracy; b=f

end
end
if Best> previous Best result
S=S ∪ b

B=B – b

mf=Best

else
break

end
n=n+1

end

Output:

s //*where s is the optimum subset of features*

mf //*where mf is the accuracy of regression from s features.*

3.1 Complexity analysis of HH-FS

In this section, the complexity analysis of the proposed HH-FS method is described in two learning and testing phases.

A) Training phase analysis

There will be three main parts including clustering, filter, and wrapper feature selections in the training phase of the proposed method. K-Means method is used in the clustering part, assuming we have M samples and C clusters, so the computational complexity will be obtained as (2):

$$O(C \times M). \quad (2)$$

Filter techniques considered ranking methods. In this category, we have a list of features that is arranged based on an assessment criterion. Thus, there are two basic actions have been done in these methods, calculating the scores and ordering them based on the gained scores. Accordingly, the time complexity of these algorithms for N features and M samples by P filter methods is as (3) [43]:

$$O(P \times (M \times N + N^2)). \quad (3)$$

Wrapper techniques are also one of the query methods in which every single feature in normal status is tested one by one. Thus, they have exponential time complexity. But in the proposed method, the number of examined cases is reduced by implementing filter methods. Therefore, assuming that we have (A) common and (B) non-common features and (X) iteration to achieve the best accuracy, equation 4 will be as (4):

$$O(X \times a) + O(X \times b). \quad (4)$$

Hence, the computational complexity of the proposed method in the training phase will be expressed in the form of (5):

$$O(C \times M(+O(P \times (M \times N + N^2)))) \\ +O(X \times a) + O(X \times b). \quad (5)$$

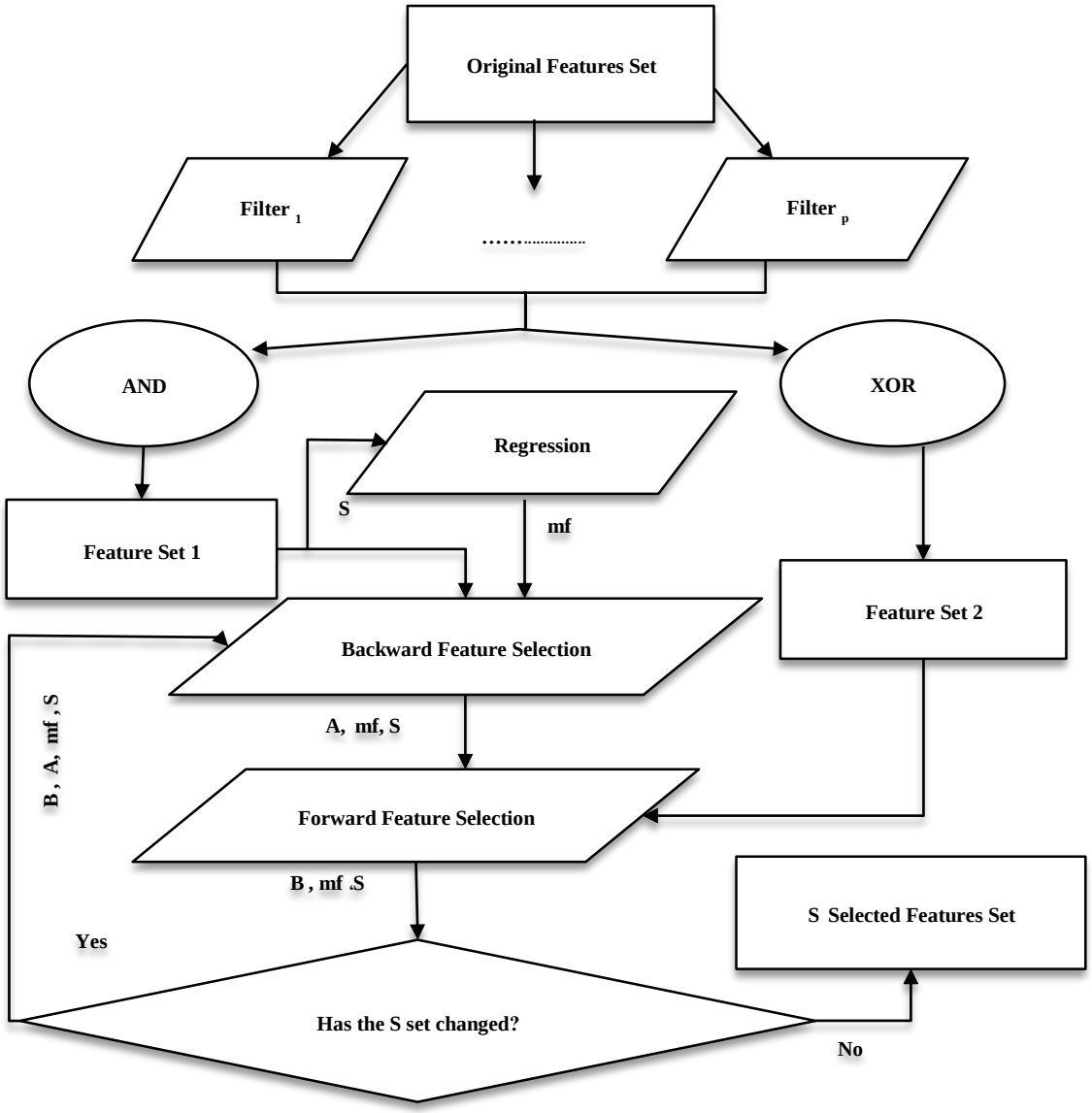

Figure 1. System architecture of the proposed Hierarchical feature selection (H-FS) method [9].

B) Testing phase analysis

In the testing phase, we first identify the cluster related to the sample and then specify its effective features and perform the estimation at the end. Thus, the similarities of the sample with cluster centers are investigated. This operation is repeated as many times as the number of clusters. Thus, the complexity for each sample will be O(C). Next, the estimation is done based on the effective features of that cluster. This is done once for each sample that is not considered in calculations. For each test sample, an ML algorithm is run. To compute the complexity in MLP, suppose there are m training samples, s features, k hidden layers, each containing h neurons for simplicity, and output neurons. Thus, the time complexity would be O(n×m×h^k×o×i), where i is the number of iterations. In this paper, o = 1 and k for four datasets are equal to one. The number of iterations for the test instance is one. For this reason, if we suppose that the number of test samples is equal to m, (6) will be obtained:

$$O(m \times C) + O(m \times s \times h). \quad (6)$$

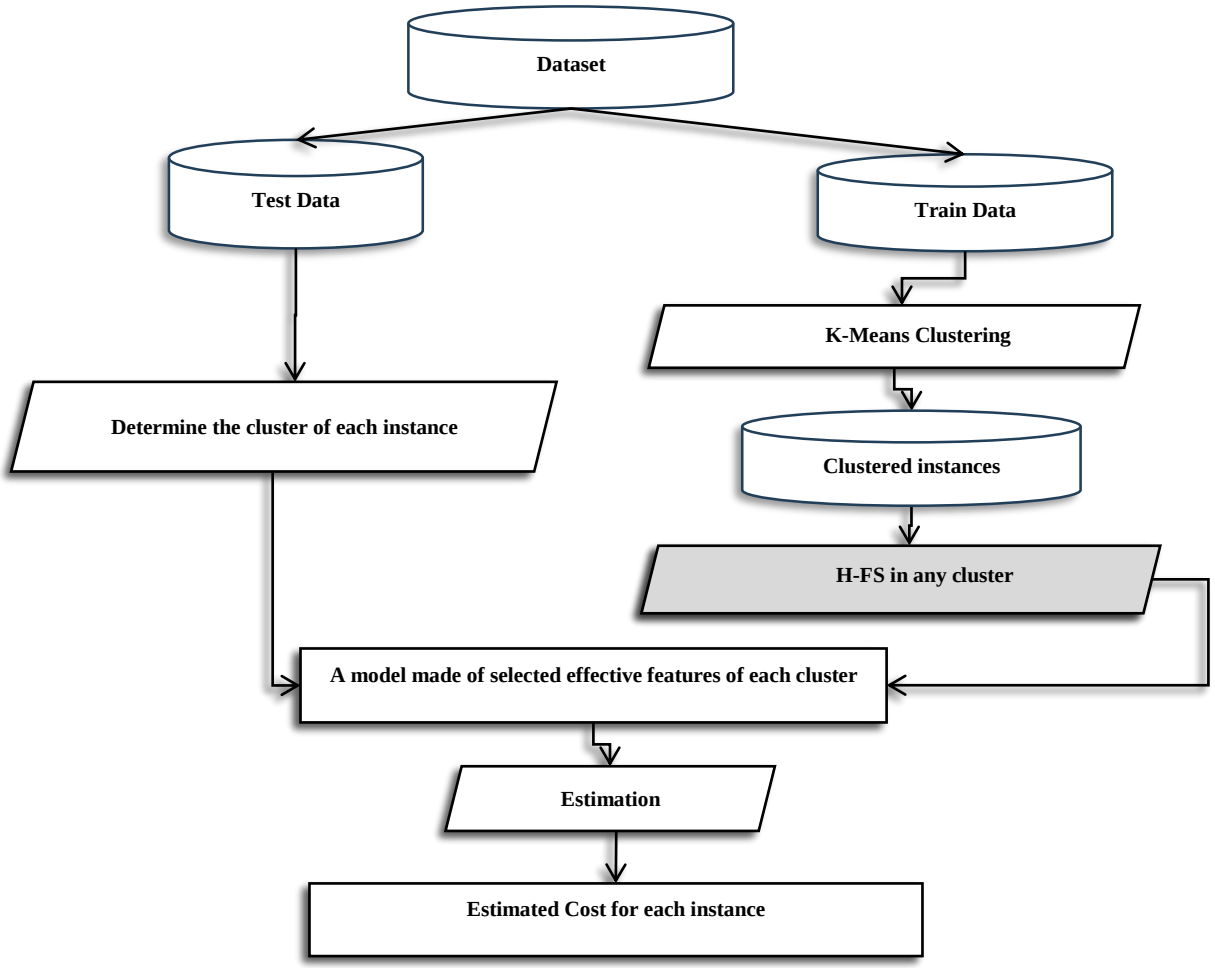

Figure 2. System architecture of the proposed Hierarchical Homogeneous Feature Selection (HH-FS) method.

4. Experimental Results

The entire process of the proposed method implementation can be observed in Figure 3. Values for each of the parameters in our algorithm in the field of software cost estimation are represented in Table 1. The parameter K is set to 2 because the EF criterion is composed of two evaluation measurements (MMRE and PRED). The number of combined filter methods in the simplest state equals 2. To determine the number of selected features in any filter method on any dataset, datasets are divided into two categories: datasets with a high number of features (higher than 10 features) and datasets with a low number of features (lower than 10 features). To set parameter T, we tried to select at least half of the features in any dataset. For this reason and based on the number of different features of each dataset, this parameter is set to 10 for the first group of datasets and is set to 4 for the second group of datasets. The entire applied algorithm in the proposed method has been implemented in MATLAB environment on a core i5, 6GB RAM personal computer with windows 7.

Table 1. Values of parameters in the proposed method.

Parameter	Value
k	2
p	2
Tp (COCOMO81)	10
Tp (COCONASA93)	10
Tp (Maxwell)	10
Tp (Desharnais)	4
Tp (Kemerer)	4
Tp (Albrecht)	4

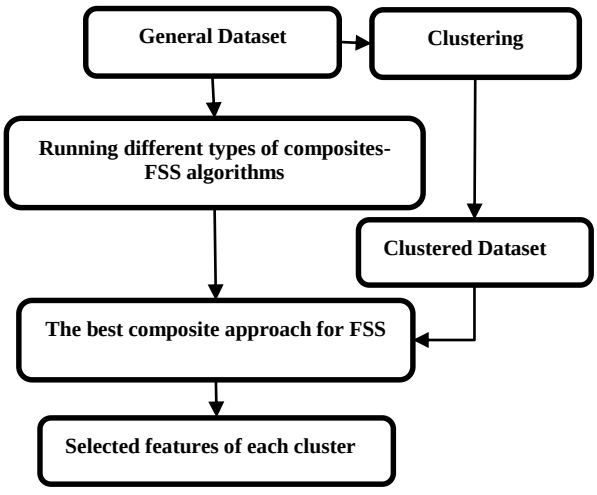

Figure 3. Chart of the proposed method implementation.

4.1. Datasets

The quality of datasets can have a significant impact on the accuracy of machine learning methods. In SCE, 13 public datasets are available, that in [8] have been studied in terms of quality. Loss of timeliness in datasets is one of the major challenges in data quality. COCOMO81, Kemerer, Albrecht, and Desharnais are four public datasets that are used more than others [48]. In Figure 4, the use of this dataset in recent articles has been reviewed. To evaluate the proposed method in the first step, the COCONASA93 and Maxwell datasets have been used along with the most widely used datasets of this field (COCOMO81, Kemerer, Albrecht, and Desharnais).The proposed method can be used even with datasets collected from projects with different methodologies and developed by different teams. In none of the datasets used to check the effectiveness of the proposed method, the methodology used is emphasized. Research in the field of software cost estimation, with emphasis on projects implemented with specific methodologies, have generally presented results based on specific datasets collected by the authors [11]. Since the data used in the field of cost estimation based on methodologies need to consider different factors, we intend to investigate the proposed method for selecting the most effective cost factors by different methodologies in future works.

4.1.1. COCOMO81

COCOMO is a regression-based model to estimate the software effort, which belongs to the category of functional algorithm models. This model was developed by Boehm in 1981 and is the most quoted, famous, and accepted algorithm model for software cost prediction. The COCOMO model can be used to calculate the amount of effort and required time for software projects. The stable COCOMO81 is one of the COCOMO models. The problem of this model is its incompatibility with environmental advances in software development. This is a problem that all algorithmic models are involved in. A dataset based on this model has been provided with the same name including 63 software described projects with 17 cost determinants ranked on a scale ranging from very low to extra high which is still used [48].

4.1.2. COCONASA93

Following the applications of feature vectors defined by Boehm et al in COCOMO, NASA represented COCONASA dataset in 2004 containing 60 software projects have been produced by various NASA centers which are all described through the same feature vector. Then in 2006, NASA introduced a new version of its dataset called COCONASA_2 with 93 project samples. This version of the NASA dataset is also known as COCONASA93. COCONASA93 is based on the COCOMO format, containing 23 cost drivers as input features [48].

4.1.3. Maxwell

This dataset includes 62 software projects that have been collected from the biggest commercial banks in Finland and it contains 26 independent variables that are determined by different software features such as application and size. Their dependent variety is the effort for software development that is specified by the number of work hours of software suppliers from technical specifications to delivery time. These dataset projects are related to years from 1985 to 1993 [48].

4.1.4. Desharnais

This dataset encompasses produced projects during the years 1981 to 1988 in a software house. The original version of this dataset contains 81 project samples have been described by 12 attributes. However, among all samples of the dataset, 4 samples have missed values in 4 features. Researchers use this dataset in different ways. A group of researchers put aside the features with missing values and some others eliminate samples with missing values from the dataset. This dataset contains 77 complete software projects and 4 incomplete samples [4].

4.1.5. Kemerer

The Kemerer dataset includes 15 software projects which were described by 5 independent attributes and one dependent attribute. The independent attributes are represented by 2 categorical and 3 numerical attributes. The effort attribute is measured by 'man-months' [48].

4.1.6. Albrecht

The Albrecht dataset contains 24 software projects that were developed by using third-generation languages such as COBOL and PL1. The dataset is described by 7 features (18) projects were written in COBOL;(4) projects were written in PL1 and the rest were written in dataset management languages [48]. The descriptive statistics of such datasets are summarized in Table 2. They often have a limited number of observations that are affected by multicollinearity and outliers. We can also observe that all the datasets have positive skewness values with range from 1.78 to 4.36. This observation indicates that the datasets are

extremely heterogeneous, which makes sure that we test the proposed model adequately.
Outliers are evident for at least one variable in all of the datasets, a finding that is consistent with prior literature on this issue has noted that outliers are a common phenomenon in the software cost estimation datasets [8].

4.2. Performance metrics

In this paper, to evaluate the accuracy of this idea in SCE, the proposed method is implemented on various datasets, and evaluation criteria of SCE are used to analyze the results. In SCE, various evaluation criteria are used. The most common used criteria are Mean Relative Error (*MRE*), which represents the difference between the estimated costs and actual costs, *MMRE*, represents the average estimation error for the total samples (training samples and test samples), and *PRED(X)*, represents the percentage of samples that their MRE is less than or equal to the value of *X*. Also in some studies, the median estimation error or *MDMRE* has been used. The description of the used formulae for the criteria that is defined above will be followed.
PRED (0.25) indicates the percent of samples with MRE less than or equal to 0.25. Therefore, the higher value of *PRED (0.25)*, denotes a less error rate of the evaluated algorithm. Thus, by picking up the features which result in lower *MMRE* and higher *PRED*, the semantic gap can be reduced in the estimation procedure.
In this paper, decision on effective feature selection in each dataset is considered as a multi-criteria decision-making technique. The weighting method was used in the present study to combine *MMRE* and *PRED* criteria; accordingly, a composite criterion was created. Furthermore, the composite criterion *EF* has already been used for ranking the machine learning algorithms. This criterion was introduced in 2009 to measure the accuracy which has been used in some studies such as [4]. Evaluation function (*EF*) is obtained from the combination of two criteria *PRED (0.25)* and *MMRE*, which is shown in Equation (7).
According to what was stated in the description of evaluation criteria, effective features are the ones that could contribute more to bridging the semantic gap between actual effort and estimated effort. This feature will be effective when selecting results in lower *MMRE* and higher *PRED*. Therefore, using a criterion that could cover both other criteria we can have both evaluation criteria in mind and try to optimize them, simultaneously. Finally, *EF* criterion was chosen as the criterion that we tended to use here due to its better results. For statistical analysis, the final results were evaluated using the following criteria.

4.3 First stage

The FEAST tool was used where a variety of filter feature selection methods were run in the MATLAB environment. It is used to run the filter procedures to score features describe software projects. There are two paths to implement filter methods on continuous feature datasets.
Some researchers would discretize the datasets and some others use kernel-based density estimators to approximate their scores. In FEAST, the continuous features are divided into five discrete bins. It should be noted that the leave-one-out cross-validation (LOOCV) method was used to separate training and test data [38]. In any dataset, one of the samples is regarded as a test sample and the rest as training samples. This operation was done as many times as the number of dataset samples and in the end, the mean error of data was reported. To choose the machine learning method which was used in this study, various parametric and non-parametric methods were examined by using the COCOMO81 dataset as an example, and finally because of providing a better accuracy, the multi-layer perceptron (MLP) was selected and used. Table 3 contains the experiment results of this evaluation with LOOCV. Machine learning techniques have parameters that affect their accuracy [39]. In this paper, to implement MLP, NETLAB MATLAB code is used. In each dataset, we tried to set MLP parameters in such a way that it is close to the primary accuracy provided in [38]. In Table 4, the MLP parameters for each dataset are presented.

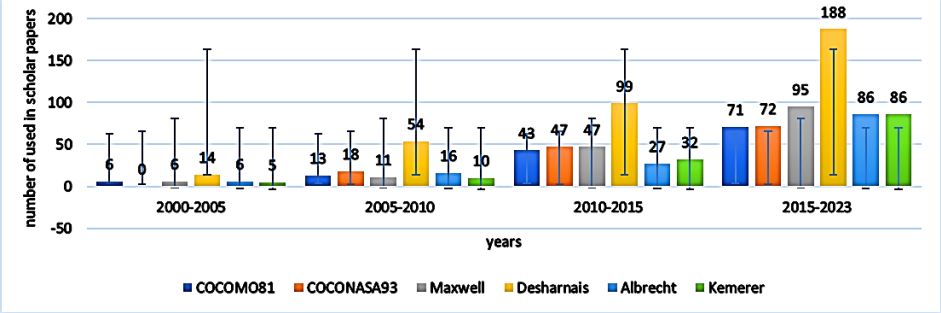

Figure 4. Review the number of used datasets in Scholar.

Table 2. Statistical properties of the employed datasets.

Dataset	Feature	Size	Effort data					
			unit	min	Max	mean	median	skew
COCOMO81	17	63	months	6	11400	683	98	4.4
COCONASA93	24	93	months	8	8211	624	252	4.2
Desharnais	11	77	hours	546	23940	5046	3647	2.0
Kemerer	8	15	months	23.2	1107.3	219.2	130.3	2.76
Albrecht	8	24	months	1	105	22	12	2.2
Maxwell	27	62	hours	583	63694	8223.2	5189.5	3.26

$$EF = PRED(0.25)/(1+MMRE). \tag{7}$$

$$AE_i = \left|Actual\ Effort - Estimated\ Effort\right|. \tag{8}$$

$$MAE = \sum_{i=1}^{n}(AE_i)/n. \tag{9}$$

$$MBRE = 1/n\sum_{1}^{n} AE_i/(min(ActualEffort_i, EstimatedEffort_i)). \tag{10}$$

$$MIBR = 1/n\sum_{1}^{n} AE_i/max(ActualEffort_i, EstimatedEffort_i). \tag{11}$$

$$RMSE = \sqrt{\sum_{i=1}^{n}(AE_i)^2/n}. \tag{12}$$

$$R^2 = 1 - \sum_{i=1}^{n}(AE_i)^2 / \sum_{i=1}^{n}(ActualEffort_i)^2, 0 \le R^2 \le 1 \tag{13}$$

Table 3. Result of parametric and non-parametric methods in COCOMO81.

Method	MMRE	MDMRE	Pred
linear Regression	20,02	20,02	9,52
Mean Smoother	1586	1586	7,93
Kernel Smoother	361	361	23,8
k-Nearest Neighbor	176,8	176,8	14,93
MLP	1.9239	0.6926	20.6349

Table 4. Parameter choice in MLP for every dataset.

Dataset	hidden units	training cycles	Coefficient of weight-decay prior
COCOMO81	1	100	0.02
COCONASA93	4	1902	0.05
Desharnais	1	100	0.02
Kemerer	1	300	0.02
Albrecht	1	100	0.02
Maxwell	1	100	0.02

In this step, different combinations of filter feature selection methods in FEAST were evaluated for various datasets based on the proposed method to select the best possible combination that was implemented in the second step. Different methods have been implemented in this tool have been described in [50] with details. Three of the best implemented combinations on six introduced datasets here along with the obtained results are represented in Table 6. In Table 6, the proposed method and its performance is compared with simple forward and backward feature selection techniques based on the *MMRE* criterion. As you can see, the proposed method is more accurate in five datasets but it provides lower performance in the Kemerer dataset than the backward selection method. These results indicate the fact that although considered measures for filtering the features in the datasets have improved the performance based on four evaluation criteria, they are less efficient than the backward method. According to different compounds, size is known as the most effective feature, which is common in all datasets. Also Table 5 indicate features that in all compounds were identified as excess features (less important one). There are not any conclusions from the results in the Kemerer dataset. We have already analyzed this step in another article with less data [9].

Table 5. Excess features of datasets.

Dataset	Excess features
COCOMO81	Cplx and Tool
COCONASA93	VIRT and VEXP
Desharnais	Transactions and Entities
Albrecht	length and Inquiry
Maxwell	T_{02} and T_{15}

4.4 Second stage

Outliers can cause the model to make incorrect assumptions about data. Outlier data is known as bad data and the purpose of identifying an outlier is to eliminate it. However, if the outliers follow meaningful patterns, appropriate methods should be used for their identification and elimination. To Handle outliers, we used data clustering instead of removing these samples. Homogeneous samples in the same clusters have effective features that is different from samples of other clusters. For this reason, we used effective feature selection individually for each cluster to increase the performance of software cost estimation. To cluster the dataset, the K-Means implemented clustering technique in MATLAB, was used. The proposed method in this study to select features in each cluster is a combined technique. Table 7 shows the results of the statistical performance evaluation of the proposed method.

Table 7 contains the results of implementation of the proposed method on six common datasets in the field of software cost estimation. The results of implementation of the proposed method on the Desharnais dataset indicate that this method operated effectively on a dataset and it could increase the prediction accuracy using this dataset. All four evaluation criteria used in the present study on the Desharnais dataset were optimized by increasing the number of clusters and it refers to this point when contradictory samples and the machine learning algorithm are separated from similar or homogeneous samples, the accuracy of these methods for predicting the required cost to produce and develop software projects will increase. Implementation of the proposed method on the COCONASA93 dataset with two clusters reduced the accuracy of the learning algorithm in compared to the initial state of datasets and selected a combined feature. However, the accuracy of the algorithm was enhanced when the dataset was divided into a greater number of clusters.

All four evaluation criteria which is used here on the COCONASA93 dataset were also optimized by increasing the number of clusters, which represent the effectiveness of the separation of contradictory samples and machine learning algorithms from the homogeneous samples on this dataset. Just like COCONASA93 with two clusters, the results of performing the proposed method on the COCOMO81 dataset also suggest that the performance of the learning algorithm has been dropped. The accuracy of the learning algorithm with two clusters has been reduced in compare to the original status of the dataset even without the composite feature. But when a dataset is divided into a higher number of clusters algorithm accuracy is increased. Though the MMRE criterion was higher with five clusters than with four, three other evaluation criteria were optimized with five clusters. All four evaluation criteria that have been used here on the COCONASA93 dataset were also

optimized by increasing the number of clusters. The reason is that since these criteria lack sensitivity to the samples, there have been few samples among all datasets with increased difference between their output and actual amount. In the end, it can be concluded that the separation of inconsistent samples and learning of machine learning algorithms from homogeneous samples has been effective in this dataset. Implementation of the proposed method on Albrecht and Kemerer datasets with two clusters has provided the best results. These two datasets have a smaller number of samples than other used datasets. The proposed method for the Kemerer dataset had less MMRE in single cluster mode. While other evaluation criteria provided more optimized results in two-cluster mode. All criteria on the Albrecht dataset in two-cluster mode represented the best obtained results from implementing the proposed method.

In the Maxwell dataset, implementing a hybrid feature selection method in states of two, three, and four clusters have less precise results than a single cluster. However, this approach retained the process of increasing accuracy by increasing the number of clusters. In five clusters mode, the proposed method provided its best accuracy. It is emphasized on this paper that at first HH-FS feature selection should be implemented for data without clustering and the best combination of feature selection methods is selected. Then the feature selection method is implemented on the datasets in states of segmenting data into two, three, four, and five clusters, and the results are expressed in each condition.

4.5. Comparison of results

The results of comparing the proposed method with obtained results in other studies are demonstrated in Table 9. The comparing studies were chosen from research conducted in recent years on six datasets used in the present paper. As seen in Table 9, results indicate that the semantic gap was reduced more using the proposed method in the present study than in comparable research. All of the compared articles used the LOOCV method for isolating training samples and tests. In [51], different methods of machine learning algorithms with backward selection, feature selection in SCE datasets are considered. Researchers in [52] investigated the use of cluster center initialization techniques to improve the estimation accuracy of the 2FA-kprototypes approach (2FA-kprototypes is based on the use of the fuzzy k-prototypes clustering technique.). CBI (Centrality-Based Initialization) and DBI (Density-Based Initialization) are two cluster-center initialization techniques that were used. In the method which is presented in [53], the Optimal projects of predicted Class and Feature Weighting and Functional Link Artificial Neural Network based Estimation (OCFWFLANN), a genetic algorithm has been used to optimize the accuracy of the Functional Link Artificial Neural Network algorithm.

In [4], researchers have used homogeneous and heterogeneous combinations of different learning methods to increase the accuracy of the estimations. In [53], a new technique to achieve the best Analogy for each sample is provided and evaluated with different adjustment techniques such as similarity-based adjustment (SM), neural network-based adjustment (NN), Genetic Algorithm based adjustment (GA) on various data in this area. In [54], the Grey relational Analysis method is used for feature selection and finding the lowest comparisons for references project among a set of projects, too. SCE and then the results with Stepwise regression (SR) and CBR are compared. A hybrid feature selection method that builds use of the advantage of supervised feature selection (features are grouped into several clusters) and unsupervised feature selection (feature in each cluster with the maximize similarity by a class label is selected as the representative feature) method is proposed in [49]. In [55], researchers have integrated analogy-based estimation by Fuzzy numbers to better the performance of software effort estimation in the early steps of a software development lifecycle. In [56], researchers have used the PSO algorithm to effort estimation and compare the results of the proposed model and the COCOMO model. They show proposed model is more accurate than the COCOMO model.

In [18], it is proposed a model consisting of data filtering and feature weighting techniques in the final step of data modeling.

5. Conclusion and suggestions

Software cost estimation is one of the important and challenging activities in software project management. Effective factors on increasing of the costs of production and development of a software product are called cost factors. The required effort to produce and develop any software product is considered the main factor in cost determination. Thus, two terms; software cost estimation and software development effort estimation are used

equivalently in this area. In this study, a new method was introduced to determine the effective cost factors among various factors offered in algorithmic models in this area.

Feature selection methods include two widely used filter and wrapper techniques. Filter methods are faster but have low accuracy. Wrapper methods apply machine learning techniques so they are highly accurate but have low speed. By combination of these two methods we can use the advantages of each one to overcome defects of the other one. These two methods were combined using a new hierarchical feature selection procedure.

The used evaluation criterion is important for measuring the effectiveness of each feature on cost prediction accuracy and required effort for developing a software product. Three evaluation criteria; MMRE, MDMRE, and PRED have been used so far in this field. Considering decisions on the effectiveness of features as a multiple-criteria decision-making process, we can choose features that could meet several criteria at the same time and improve them, simultaneously. EF criterion which is resulted from the combination of two MMRE and PRED criteria has been used to rank operated algorithms in software cost estimation. This criterion was used as an evaluation measure to assess the effectiveness of cost factors. The innovation of this paper was selection of the effective features for similar samples. Training data in SCE usually contain outlier and contradictory samples. We can reduce the effect of contradictory and outlier datasets on training machine learning techniques by separating homogeneous samples from heterogeneous ones. To this end, data clustering was performed to reduce the impact of contradictory samples, and the hierarchical feature selection technique was implemented on clustered samples with compound criteria introduced here on. In this article, the K-Means clustering method was used for homogenization. Other methods such as K-Means ++ clustering can be used in future works. To evaluate the proposed method, it was implemented on five well-known and common used datasets in the field of software cost estimation and the results were demonstrated. The empirical obtained results indicate the effectiveness of this method for the determination of effective cost factors. In this study, this method was executed in two phases. In the first step, the best possible combination was run on datasets without clusters and in the second step the best specified combination was applied on clustered datasets. According to the results of both steps, we can conclude that the use of clustering has reduced the effect of contradictory and outlier samples on the generalization of the machine learning method used in this paper.

From the beginning until now, several machine learning techniques have been used in conducted research on software cost estimation. Using new algorithms such as extended versions of SVR can increase the accuracy. Since there are several evaluation criteria in the field of software cost estimation, a new measure can be provided by combining existing criteria using multi-criteria decision-making techniques to select the most effective cost factors. In this study, a combination of two filter methods along with two wrapper methods was used to select effective features. It seems that more of these techniques are combined, the accuracy of the model and feature selection methods is increased. Filter methods have different selection criteria. Considering this, combining criteria from different filter families can positively affect selecting the most effective features. In this study, backward and forward features section methods were used as representatives of wrapper techniques. The problem of these methods is that they are likely stopped at local optima. Using other wrapper techniques such as heuristic methods may be effective in improving the accuracy of learning methods.

In the future, we will have an increment in management software systems with a diversity of data-driven prediction features. The estimation of costs at different levels of complexity, ranging from individual software stories up to whole modules or projects will be one of these features. Thus, the number of software companies adopt an agile software development methodology will increase, and cost estimation methods focusing on tracking individual engineers' development patterns will be developed.

Examining the proposed method in agile methodology is one of the tasks that we will focus on in the future.

Table 6. Results obtained from the first stage.

Dataset	Ranks	Filter composite	MMRE	MDMRE	PRED	EF
COCOMO81	Only neural network		1.9239	0.6926	20.6349	7.0572
	1	Betagamma,Relife	1.6352	0.7146	**28.5714**	**10.8421**
	2	CIEF, ICAP	1.6753	**0.5807**	**28.5714**	10.6797
	3	CIEF,condred	1.6753	**0.5807**	**28.5714**	10.6797
	4	Forward based on *MMRE*	**1.1671**	0.6331	20.6349	9.5218
	5	Backward based on *MMRE*	1.9239	0.6926	20.6349	7.0572
COCONASA93	Only neural network		1.2484	0.4193	30.1075	13.3908
	1	mRMR, CIEF	1.1865	**0.3398**	**44.0860**	**20.1627**
	2	Diser,CIEF	1.1808	0.3773	41.9355	19.2298
	3	Diser,condred	1.1444	0.3560	40.8602	19.0547
	4	Backward based on *MMRE*	1.2484	0.4193	30.1075	13.3908
	5	Forward based on *MMRE*	**1.0404**	0.4682	23.6559	11.5940
Maxwell	Only neural network		0.6639	0.5611	25.8065	15.5100
	1	Icap, Betagamma	0.4465	**0.2447**	**53.2258**	**36.7958**
	2	mrme,icap	0.4465	**0.2447**	**53.2258**	**36.7958**
	3	icap,condred	0.4465	**0.2447**	**53.2258**	**36.7958**
	4	Backward based on *MMRE*	0.5284	0.4037	27.4194	17.9396
	5	Forward based on *MMRE*	**0.4093**	0.2976	41.9355	29.7564
Kemerer	Only neural network		0.7245	0.3924	26.6667	15.4637
	1	Jmi,icap	0.4238	0.3241	40.0000	28.0931
	2	Jmi,diser	0.4238	0.3241	40.0000	28.0931
	3	Relief,jmi	0.4238	0.3241	40.0000	28.0931
	4	Forward based on *MMRE*	0.4578	**0.2840**	40.0000	27.4384
	5	Backward based on *MMRE*	**0.3888**	0.2975	**46.6667**	**33.6030**
Albrecht	Only neural network		1.6309	0.6123	33.3333	12.6700
	1	Mifs, Cife	**0.7603**	0.5201	**41.6667**	**23.6701**
	2	Mifs,icap	**0.7603**	0.5201	**41.6667**	**23.6701**
	3	Mifs, relief	0.7952	**0.3922**	37.5000	20.8894
	4	Backward based on *MMRE*	1.6309	0.6123	33.3333	12.6700
	5	Forward based on *MMRE*	0.7952	**0.3922**	37.5000	20.8894
Desharnais	Only neural network		0.9067	0.4574	28.3951	14.8923
	1	Mifs, Relife	0.6182	**0.3245**	**40.7407**	**25.1773**
	2	mRMR,Relife	0.6527	0.3437	**40.7407**	24.6516
	3	CIEF,Relife	0.6527	0.3437	**40.7407**	24.6516
	4	Forward based on *MMRE*	**0.5650**	0.3779	35.8025	22.8769
	5	Backward based on *MMRE*	**0.5650**	0.3912	32.0988	20.5639

Table 7. Results of Statistical evaluation in implementing the proposed method on datasets.

dataset	MAE	RMSE	MIBR	MIBRE	R2
COCOMO81	234.4313	871.9992	0.6670	0.3583	0.7963
COCONASA93	140.7556	475.2479	0.1234	0.2246	0.8645
maxwell	2057.8091	4949.6150	0.2337	0.1930	0.8609
desharnais	808.7592	1349.9374	0.2173	0.1726	0.9593
Albrecht	4.4387	6.5195	0.4245	0.2579	0.9661
kemerer	114.5817	239.5492	0.4898	0.3290	0.4906

Table 8. Results of implementing the proposed method on datasets.

data	step	MMRE	MDMRE	PRED	EF
COCOMO81	**MLP**	1.9239	0.6926	20.6349	7.0572
	1 Cluster	1.6352	0.7146	28.5714	10.8421
	2 Cluster	1.1219	0.7303	14.2857	6.7325
	3 Cluster	1.2710	0.5406	25.3968	11.1831
	4 Cluster	**0.6467**	0.4327	31.7460	19.2784
	5 Cluster	0.6489	**0.3486**	**41.2698**	**25.0287**
COCONASA 93	**MLP**	1.2484	0.4193	30.1075	13.3908
	1 Cluster	1.1865	0.3398	44.0860	20.1627
	2 Cluster	1.3058	0.4588	34.4086	14.9225
	3 Cluster	9.7162	0.3619	45.1613	26.3148
	4 Cluster	0.5778	0.2461	51.0870	32.3793
	5 Cluster	**0.3129**	**0.1874**	**62.3656**	**47.5030**
Maxwell	**MLP**	0.6639	0.5611	25.8065	15.5100
	1 Cluster	0.4465	0.2447	53.2258	36.7958
	2 Cluster	0.4420	0.3414	40.3226	27.9624
	3 Cluster	0.3887	0.2983	43.5484	31.3595
	4 Cluster	0.3297	0.2692	48.3871	36.3904
	5 Cluster	**0.2284**	**0.1891**	**64.5161**	**52.5200**
Desharnais	**MLP**	0.9067	0.4574	28.3951	14.8923
	1 Cluster	0.6182	0.3245	40.7407	25.1773
	2 Cluster	0.5010	0.2556	49.3827	32.8997
	3 Cluster	0.4182	0.2457	54.3210	38.3025
	4 Cluster	0.3344	0.2073	64.1975	48.1101
	5 Cluster	**0.2173**	**0.1712**	**79.0123**	**64.9077**
Kemerer	**MLP**	0.7245	0.3924	26.6667	15.4637
	1 Cluster	**0.4238**	0.3241	40.0000	28.0931
	2 Cluster	0.4898	**0.2597**	**46.6667**	**31.3243**
	3 Cluster	0.6238	0.2972	33.3333	20.5279
	4 Cluster	0.9040	0.4641	20	10.5039
	5 Cluster	0.7252	0.4204	21.4286	12.4208
Albrecht	**MLP**	1.6309	0.6123	33.3333	12.6700
	1 Cluster	0.7603	0.5201	41.6667	23.6701
	2 Cluster	**0.4245**	**0.2319**	**58.3333**	**40.9515**
	3 Cluster	0.9484	0.3047	41.6667	21.3852
	4 Cluster	0.8922	0.3383	41.6667	22.0201
	5 Cluster	0.8447	0.3035	45.8333	24.8459

Table 9. Results comparison with other research.

Dataset	Research	MMRE	MDMRE	PRED	EF
COCOMO81	**This paper**	**0.65**	**0.35**	**41.27**	**25.03**
	MLP_BFS [38]	-	0.79	17.5	-
	CART+BFS [38]	-	0.65	12.7	-
	2FA-kprototypes+CBI [41]	-	-	19.05	-
	2FA-kprototypes+DBI [41]	-	-	20.63	-
	2FA-kprototypes+CBI_CAV [41]	-	-	22.22	-
	2FA-kprototypes+CBI_WM [41]	-	-	22.22	-
	SVR [58]	2.9086	-	12.5	3.19
	MLP [58]	19.1756	-	6.25	0.31
	FEEM [18]	-	0.53	0.19	-
	Linear Regression [59]	17.82	-	6	0.32
	Multilayer Perceptron [59]	5.22	-	9	1.45
COCONASA93	**This paper**	0.31	**0.19**	**62.37**	**47.50**
	MLP+BFS [38]	-	0.38	37.6	-
	OCFWFLANN [42]	**0.27**	0.19	26	20.47
	CART+BFS [38]	-	0.45	28	-
	SVR [58]	2.25	-	8.33	2.56
	MLP [58]	2.50	-	12.5	3.57
	Linear Regression [59]	1.8	-	19	6.78
	Multilayer Perceptron [59]	2.86	-	20	5.18
Maxwell	**This paper**	**0.2284**	**0.1891**	**64.52**	**52.52**
	MLP+BFS [38]	-	0.44	23.1	-
	CART+BFS [38]	-	0.45	30.4	-
	Gravitational-based KRR with RBF Kernel [60]	0.34	-	43.29	32.42
	Gravitational-based KRR with Wavelet Kernel [60]	0.35	-	44.86	33.25
	McFIS [61]	0.48	-	74.06	5041
	FEEM [18]	-	0.24	0.5	-
Desharnais	**This paper**	**0.22**	**0.17**	**79.01**	**64.91**
	CART+BFS [38]	-	0.30	35.8	-
	mRMR FS [49]	0.4325	-	44	30.72
	INMIFS [49]	0.4644	-	44.37	30.30
	FSFC [49]	0.7425	-	36.25	20.80
	HFSFC [49]	0.3406	-	50.62	37.76
	Gravitational-based KRR with RBF Kernel [60]	0.2489	-	75.82	60.71
	Gravitational-based KRR with Wavelet Kernel [60]	0.2145	-	78.24	64.42
	Open Hybrid [62]	0.26	0.17	0.59	46.83
	SVR [58]	1.30	-	28.57	12.42
	MLP [58]	1.74	-	28.57	10.42
	FEEM [18]	-	0.22	51	-
	PSO-SA + WEue [63]	0.38	0.20	38	27.53
	PSO-SA + WMan [63]	0.38	0.24	50	36.23
	PSO-SA +WMink [63]	0.49	0.34	38	25.50
Albrecht	**This paper**	0.4245	**0.23**	**58.33**	**40.95**
	KBE+FS [42]	**0.275**	-	48.7	38.2
	k-ABESM +FS [42]	0.50	-	20.8	13.87
	k-ABEGA+FS [42]	0.643	-	29.2	17.77
	k-ABENN+FS [42]	0.579	-	25.0	15.83
	GFNSE [55]	0.50	-	50.00	33.33
	CBR [55]	0.635	0.389	33.3	20.37
	Stepwise Regression [55]	0.6124	0.323	37.5	23.26
	Gravitational-based KRR with RBF Kernel [60]	0.7322	-	62.50	36.07
	Gravitational-based KRR with Wavelet Kernel [60]	0.6223	-	72.83	43.61
	SVR [58]	0.88	-	16.66	8.86
	MLP [58]	0.69	-	50	29.58
	PSO-SA + WEue [63]	0.94	0.35	25	12.89
	PSO-SA + WMan [63]	0.84	0.25	50	27.17
	PSO-SA +WMink [63]	0.52	0.39	50	32.90
Kemerer	**This paper**	0.4898	**0.26**	**46.66**	**31.32**
	KBE+FS [42]	**0.31**	-	36.7	27.80
	k-ABESM +FS [42]	0.44	-	30.0	20.83
	k-ABEGA+FS [42]	0.50	-	26.7	17.68
	k-ABENN+FS [42]	0.67	-	13.3	7.96
	GRA+ feature extraction [54]	0.62	-	-	-
	GRA+ MLR+ feature extraction [54]	0.53	-	-	-
	GRA+ stepwise+ feature extraction [54]	0.53	-	-	-
	PSO [56]	0.5657	-	-	-
	SVR [58]	1.91	-	21.60	7.42
	MLP [58]	1.09	-	25.0	11.96

Reference

[1] S. K. Sehra, Y. S. Brar, N. Kaur and S. S. Sehra, "Research patterns and trends in software effort estimation," *Information and Software Technology*, vol. 91, pp. 1-21, 2017.

[2] S. Sarwar and M. Gupta, "Proposing effort estimation of cocomo ii through perceptron learning rule," *International Journal of Computer Applications*, vol. 70, no. 1, 2013.

[3] P. Pandey, "Analysis of the techniques for software cost estimation," in *2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT)*, 2013.

[4] M. O. Elish, T. Helmy and M. I. Hussain, "Empirical study of homogeneous and heterogeneous ensemble models for software development effort estimation," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[5] E. Papatheocharous, H. Papadopoulos and A. S. Andreou, "Feature subset selection for software cost modelling and estimation," *arXiv preprint arXiv:1210.1161, 2012*.

[6] C. Kirsopp, M. J. Shepperd and J. Hart, "Search heuristics, case-based reasoning and software project effort prediction," *in the 4th Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc*, 2002.

[7] T. Menzies, K. Ammar, A. Nikora and J. DiStefano, "How simple is software defect detection," *Submitted to the Emprical Software Engineering Journa*l, 2003.

[8] M. F. Bosu, "Data quality in empirical software engineering: An investigation of time-aware models in software effort estimation (Doctoral dissertation, University of Otago)," *Doctoral dissertation, University of Otago*, 2016.

[9] S. Beiranvand and Z. Chahooki, "Bridging the semantic gap for software effort estimation by hierarchical feature selection techniques," *Journal of AI and Data Mining*, vol. 4, no. 2, pp. 157-168, 2016.

[10] Uc-Cetina and V´ıctor, "Recent Advances in Software Effort Estimation using Machine Learning," *arXiv preprint arXiv:2303.03482* (2023).

[11] C. A. P. Rodríguez, L. M. S. Martínez, D. H. P. Ordóñez and J. A. T. Peña, "Effort Estimation in Agile Software Development: A Systematic Map Study," *INGE CUC*, vol. 19, no. 1, 2023.

[12] J. Antil and R. Rishi, "SOFTWARE COST ESTIMATION USING TEMPORAL DATA MINING TECHNIQUES: AN OVERVIEW," *Journal of Data Acquisition and Processing*, vol. 38, no. 2, pp. 2718-2728, 2023.

[13] Usman, Muhammad, J. Börstler and K. Petersen, "An effort estimation taxonomy for agile software development," I*nternational Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 4, pp. 641-674, 2017.

[14] A. Zaid, M. H. Selamat, A. Ghani, R. Atan and K. Wei, "Issues in Software Cost Estimation," *International Journal of Computer Science and Network Security*, vol. 8, no. 11, pp. 350-356, 2008.

[15] T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation.," *Swarm and Evolutionary Computation*, vol. 38, pp. 158-172, 2018.

[16] J. Wen, S. Li, Z. Lin, Y. Hu and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information and Software Technology*, vol. 54, no. 1, pp. 41-59, 2012.

[17] S. M. R. Chirra and H. Reza, "A survey on software cost estimation techniques," *Journal of Software Engineering and Applications*, vol. 12, no. 6, pp. 226-248, 2019.

[18] A. Moradbeiky, "FEEM: A Flexible Model based on Artificial Intelligence for Software," *Journal of Artificial Intelligence and Data Mining (JAIDM)*, vol. 11, no. 1, pp. 39-51, 2023.

[19] B. Baskeles, B. Turhan and A. Bener, "Software effort estimation using machine learning methods," *in 2007 22nd international symposium on computer and information sciences*, IEEE, 2007.

[20] L. Radlinski, "A survey of bayesian net models for software development effort prediction," *International Journal of Software Engineering and Computing*, vol. 2, no. 2, pp. 95-109, 2010.

[21] A. Chavoya, C. Lopez-Martin, I. R. Andalon-Garcia and M. Meda-Campaña, "Genetic programming as alternative for predicting development effort of individual software projects," *PloS one*, vol. 7, no. 11, p. e50531, 2012.

[22] V. Venkataiah, R. Mohanty and M. Nagaratna, "Prediction of software cost estimation using spiking neural networks," *in Smart Intelligent Computing and Applications: Proceedings of the Second International Conference on SCI 2018*, Volume 2, Springer, 2019.

[23] S. Bibi, I. Stamelos and L. Angelis, "Combining probabilistic models for explanatory productivity estimation," *Information and Software Technology*, vol. 50, no. 7-8, pp. 656-669, 2008.

[24] M. Hosni, A. Idri and A. Abran, "Evaluating filter fuzzy analogy homogenous ensembles for software development effort estimation," *Journal of Software: Evolution and Process*, vol. 31, no. 2, p. e2117, 2019.

[25] O. Jalali, T. Menzies, D. Baker and J. Hihn, "Column pruning beats stratification in effort estimation," *in Third International Workshop on Predictor Models in Software Engineering (PROMISE'07: ICSE Workshops 2007)*, 2007.

[26] M. A. Saleem, R. Ahmad, T. Alyas, M. Idrees, A. Farooq, A. S. Khan and K. Ali, "Systematic literature review of identifying issues in software cost estimation techniques," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 8, pp. 341-346, 2019.

[27] M. A. Christina and C. Banumathy, "Software cost estimation using neuro fuzzy logic Framework," *International Journal of Research in Engineering, Science and Management*, vol. 2, no. 1, pp. 219-224, 2019.

[28] H. Mustapha and N. Abdelwahed, "Investigating the use of random forest in software effort estimation," *Procedia computer science*, vol. 148, pp. 343-352, 2019.

[29] Z. Chen, T. Menzies, D. Port and B. Boehm, "Feature subset selection can improve software cost estimation accuracy," *in Proceedings of the 2005 workshop on Predictor Models in Software Engineering*, 2005.

[30] Q. Song, J. Ni and G. Wang, "A fast clustering-based feature subset selection algorithm for high-dimensional data," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 1, pp. 1-14, 2011.

[31] E. Kocaguneli, T. Menzies, J. Keung, D. Cok and R. Madachy, "Active learning and effort estimation: Finding the essential content of software effort estimation data," *IEEE Transactions on software engineering*, vol. 39, no. 8, pp. 1040-1053, 2012.

[32] T. Menzies, D. Port, Z. Chen and J. Hihn, "Specialization and extrapolation of software cost models," *in Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, ACM, 2005.

[33] S.-J. Huang and N.-H. Chiu, "Optimization of analogy weights by genetic algorithm for software effort estimation," *Information and software technology*, vol. 48, no. 11, pp. 1034-1045, 2006.

[34] J. W. Keung, B. A. Kitchenham and D. R. Jeffery, "Analogy-X: providing statistical inference to analogy-based software cost estimation," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 471-484, 2008.

[35] M. Auer, A. Trendowicz, B. Graser, E. Haunschmid and S. Biffl, "Optimal project feature weights in analogy-based cost estimation: Improvement and limitations," *IEEE Transactions on Software Engineering*, vol. 32, no. 9, pp. 83-92, 2006.

[36] T. S. Sethi, C. V. Hari, B. Kaushal and A. Sharma, "Cluster analysis & Pso for software cost estimation," *in Information Technology and Mobile Communication: International Conference, AIM 2011, Nagpur, Maharashtra*, Springer Berlin Heidelberg, 2011.

[37] M. J. Madari and M. Niazi, "Improve software effort estimation using information entropy," *International Journal of Computer Science Issues (IJCSI)*, vol. 16, no. 2, pp. 17-22, 2019.

[38] L. L. Minku and X. Yao, "Ensembles and locality: Insight on improving software effort estimation," *Information and Software Technology*, vol. 55, no. 8, pp. 1512-1528, 2013.

[39] Huang, Sun-Jen, N.-H. Chiu and Y.-J. Liu, "A comparative evaluation on the accuracies of software effort estimates from clustered data," *Information and Software Technology*, vol. 50, no. 9-10, pp. 879-888, 2008.

[40] A. Sharma and N. Chaudhary, "Prediction of Software Effort by Using Non-Linear Power Regression for Heterogeneous Projects Based on Use case Points and Lines of code," *Procedia Computer Science*, vol. 218, p. 1601–1611, 2023.

[41] S. Hameed, Y. Elsheikh and M. Azzeh, "An Optimized Case-Based Software Project Effort Estimation Using Genetic Algorithm," *Information and Software Technology*, vol. 153, p. 107088, 2023.

[42] S. Hameed, Y. Elsheikh and M. Azzeh, "An Optimized Case-Based Software Project Effort Estimation Using Genetic Algorithm," *Information and Software Technology*, vol. 153, p. 107088, 2023.

[43] S. S. ALI, J. REN, K. ZHANG, J. WU and C. LIU, "Heterogeneous Ensemble Model to Optimize Software Effort Estimation Accuracy," *IEEE Access*, vol. 11, pp. 27759-27792, 2013.

[44] J. Aroba, J. J. Cuadrado-Gallego, M.-Á. Sicilia, I. Ramos and E. Garcia-Barriocanal, "Segmented software cost estimation models based on fuzzy clustering," *Journal of Systems and Software*, vol. 81, no. 11, pp. 1944-1950, 2008.

[45] B. Ghotra, S. McIntosh and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," *in 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, IEEE, IEEE.

[46] T. S. Sethi, C. Hari, B. Kaushal and A. Sharma, "Cluster Analysis and Pso for Software Cost Estimation," *in Information Technology and Mobile Communication: International Conference*, AIM 2011, Nagpur, Maharashtra, India, April 21-22, 2011. Proceedings. Springer Berlin Heidelberg, 2011., Springer, 2011.

[47] J. Rodriguez, L. Kuncheva and C. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 10, pp. 1619-1630, 2006.

[48] P. Nerurkara, A. Shirkeb, M. Chandanec and S. Bhirud, "Empirical Analysis of Data Clustering Algorithms," *in 6th International Conference on Smart Computing and Communications*, Kurukshetra, India, 2017.

[49] V. Khatibi, D. N. Jawawi and E. Khatibi, "Increasing the accuracy of analogy based software development effort estimation using neural networks," *International Journal of Computer and Communication Engineering*, vol. 2, no. 1, p. 78, 2013.

[50] K. Dejaeger, W. Verbeke, D. Martens and B. Baesens, "Data mining techniques for software effort estimation: a comparative study," *IEEE transactions on software*, vol. 38, no. 2, pp. 375-397, 2011.

[51] A. C. Pocock, "Feature selection via joint likelihood," *Doctoral dissertation, University of Manchester*, 2012.

[52] V. K. Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim and E. Khatibi, "Increasing the accuracy of software development effort estimation using projects clustering," *IET software*, vol. 6, no. 6, pp. 461-473, 2012.

[53] Y.-S. K.-A. Y. a. D.-H. B. Seo, "An empirical analysis of software effort estimation with outlier elimination," *in Proceedings of the 4th international workshop on Predictor models in software engineering*, 2008.

[54] T. R. Benala, S. Dehuri, S. C. Satapathy and S. Madhurakshara, "Genetic algorithm for optimizing functional link artificial neural network based software cost estimation," *in Proceedings of the international conference on information systems design and intelligent applications*, 2012.

[55] G. Nagpal, M. Uddin and A. Kaur, "A hybrid technique using grey relational analysis and regression for software effort estimation using feature selection," *International Journal of Soft Computing and Engineering*, vol. 1, no. 6, pp. 20-27, 2012.

[56] M. Azzeh, D. Neagu and P. I. Cowling, "Analogy-based software effort estimation using Fuzzy numbers," *Journal of Systems and Software*, vol. 84, no. 2, pp. 270-284, 2011.

[57] A. Kumar, B. Patro and B. K. Singh, "Parameter Tuning for Software Effort Estimation Using Particle Swarm Optimization Algorithm," I*nt. J. Appl. Eng. Res*, vol. 14, no. 2, pp. 139-144, 2019.

[58] I. U. Rehman, Z. Ali and Z. Jan, "An Empirical Analysis on Software Development Efforts Estimation in Machine Learning Perspective," *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 10, no. 3, pp. 227-240, 2021.

[59] V. Resmi and K. Anitha, "Software Effort Estimation using Machine Learning Techniques," *A Journal of Physical Sciences, Engineering and Technology*, vol. 15, no. 01, pp. 86-90, 2023.

[60] M. B. Dowlatshahi, M. A. Zare-Chahooki, S. Beiranvand and A. Hashemi, "GKRR: A gravitational-based kernel ridge regression for software development effort estimation," *Journal of Mahani Mathematical Research*, vol. 11, no. 3, pp. 147-174, 2022.

[61] E. Praynlin, "Using meta-cognitive sequential learning Neuro-fuzzy inference system to estimate software development effort," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 9, pp. 8763-8776, 2021.

[62] A. Moradbeiky, V. K. Bardsiri and M. Jafari, "Open Hybrid Model: A New Ensemble Model for Software Development Cost Estimation," *Computing and Informatics*, vol. 39, no. 6, pp. 1148-1171, 2020.

[63] Z. shahpar, V. K. Bardsiri and A. K. Bardsiri, "Hybrid PSO-SA approach for feature weighting in analogy-based software project effort estimation," *Journal of AI and Data Mining*, vol. 9, no. 3, pp. 329-340, 2021.

[64] M. U. A. K. Geeta Nagpal, "A Hybrid Technique using Grey Relational Analysis and Regression for Software *Computing and Engineering*, vol. 1, no. 6, pp. 20-27, 2012.

[65] B. P. B. K. S. Ashok Kumar, "Parameter Tuning for Software Effort Estimation Using Particle Swarm Optimization Algorithm," *International Journal of Applied Engineering Research*, vol. 14, no. 2, pp. 139-144, 2019.

[66] M. A. I. a. A. A. Hosni, "Evaluating filter fuzzy analogy homogenous ensembles for software development effort estimation," *Journal of Software: Evolution and Process*, vol. 31, no. 2, p. e2117, 2019.

[67] Z. Chen, T. Menzies, D. Port and D. Boehm, "Finding the right data for software cost modeling," *IEEE software*, vol. 22, no. 6, pp. 38-46, 2005.

[68] A. Sharma and N. Chaudhary, "Prediction of Software Effort by Using Non-Linear Power Regression for Heterogeneous Projects Based on Use case Points and Lines of code," *Procedia Computer Science 218*, pp. 1601-1611, 2023.

افزایش دقت تخمین هزینه‌ی نرم افزار مبتنی بر انتخاب ویژگی‌های مرتبط در خوشه های همگن

صبا بیرانوند[1,*] و محمدعلی زارع چاهوکی[2]

[1] گروه مهندسی کامپیوتر، دانشگاه فنی و حرفه ای، تهران، ایران.

[2] گروه مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران.

ارسال ۲۰۲۳/۰۳/۰۱؛ بازنگری ۲۰۲۳/۰۵/۰۱؛ پذیرش ۲۰۲۳/۰۸/۰۱

چکیده:

تخمین هزینه نرم‌افزار (SCE) یکی از پرکاربردترین و موثرترین فعالیت‌ها در مدیریت پروژه است. برخی از ویژگی‌ها ، اثرات نامطلوبی بر دقت در روش‌های یادگیری ماشین دارند. بنابراین، روش‌های پیش‌پردازش مبتنی بر کاهش ویژگی‌های غیرموثر می‌تواند دقت این روش‌ها را بهبود بخشد. در روش‌های خوشه‌بندی، نمونه‌ها بر اساس شباهت معنایی دسته‌بندی می‌شوند. براین‌اساس، در مطالعه پیشنهادی، برای بهبود دقت SCE، ابتدا نمونه‌ها براساس ویژگی‌های اصلی خوشه‌بندی می‌شوند. سپس برای هر خوشه، یک روش انتخاب ویژگی (FS) به طور جداگانه اجرا می‌شود. روش FS پیشنهادی، مبتنی بر ترکیبی از روش‌های FS فیلتر و بسته‌بندی است. روش پیشنهادی از مزایای هردو روش در انتخاب ویژگی‌های موثر هر خوشه، با پیچیدگی محاسباتی کمتر و دقت بیشتر استفاده می‌کند. علاوه‌براین، ازآنجایی که معیارهای ارزیابی تأثیرات قابل توجهی بر روش‌های بسته‌بندی دارند، از یک معیار ترکیبی نیز استفاده شده است. روش پیشنهادی با استفاده از دادگان Desharnais، COCOMO81، COCONASA93، Kemerer و Albrecht مورد بررسی قرار گرفت و میانگین بزرگی خطای نسبی به‌دست‌آمده (MMRE) به‌ترتیب برابر ۰,۲۱۷۳، ۰,۶۴۸۹، ۰,۳۱۲۹، ۰,۴۸۹۴، ۰,۴۲۴۵ بود. مقایسه نتایج بدست آمده با مطالعات قبلی نشان دهنده‌ی بهبود در میزان خطای SCE است.

کلمات کلیدی: تخمین هزینه نرم افزار (SCE)، تخمین تلاش نرم افزار (SEE)، روش‌های یادگیری ماشین، خوشه بندی، و انتخاب ویژگی.