Journal of AI and Data Mining
Vol 4, No 2, 2016, 157-168

10.5829/idosi.JAIDM.2016.04.02.04

Bridging the semantic gap for software effort estimation by hierarchical feature selection techniques

S. Beiranvand* and M. A. Z. Chahooki

Electrical & Computer Engineering Department, Yazd University, Yazd, Iran.

Received 27 June 2015; Accepted 06 April 2016
**Corresponding author: saba.beiranvand@stu.yazd.ac.ir (S.Beiranvand).*

Abstract
Software project management is one of the significant activates in the software development process. Software development effort estimation (SDEE) is a challenging task in the software project management. SDEE has been an old activity in computer industry from 1940s, and thus it has been reviewed for several times. A SDEE model is appropriate if it provides the accuracy and confidence simultaneously before a software project contract. Due to the uncertain nature of development estimates, and in order to increase the accuracy, researchers have recently focused on machine learning techniques. Choosing the most effective features to achieve higher accuracy in machine learning is crucial. In this work, for narrowing the semantic gap in SDEE, a hierarchical filter and wrapper feature selection (FS) techniques and fused measurement criteria are developed in a two-phase approach. In the first phase, the two-stage filter FS methods provide start sets for the wrapper FS techniques. In the second phase, a fused criterion is proposed to evaluate the accuracy in wrapper FS techniques. The experimental results show the validity and efficiency of the proposed approach for SDEE over a variety of standard datasets.

Keywords: *Software Development Effort Estimation (SDEE), Software Cost Estimation (SCE), Machine Learning (ML), Hierarchical Feature Selection (FS).*

1. Introduction

Software project management is the most important activity in any software engineering methodology. SCE for development and maintenance processes in software engineering is a challenging activity, on which many researches have focused. Similarly, SDEE is the process of effort prediction for software system development. SDEE includes software development and maintenance efforts. In software engineering researches, cost and effort estimation are used equivalently [1-4].

Accurate estimation of development cost has an important role in the success or failure of a software project. Algorithmic methods, expert judgment, and ML techniques are the general approaches in these area. Algorithmic methods are only based on the old data. Therefore, advances in software engineering are not considered in them. With regard to the rapid advancement in these areas, new effective features are recognized. Also in order to investigate the various feature effects, constant and fixed methods are not sufficient. Since algorithmic methods use a constant proven formula to calculate the software cost, these new feature effects on the system performance cannot be evaluated.

Expert judgment methods are applied by experts in a particular organization. Hence, the same accuracy in other organizations is not provided by them. Due to the uncertainty in estimating software cost, using uncertain and flexible machine learning techniques plays an important role in accuracy improvement in SDEE. The ability to perform intelligent computational methods for modeling complex set of relationships between effort and influencing factors and also their ability to learn from the old project data are the main advantages of the ML methods [5].

SDEE is an old process that began simultaneously with the computer industry in the 1940s [5]. In 1980, many developments were introduced on its

models and techniques [6]. In the same year, Boehm et al. modified the COCOMO model previously developed by them. The result obtained was a new model called COCOMOII. From 1990s onwards, extensive researches were carried out for improvement of software industry and information technology [6]. Categorization of cost estimation methods is represented in figure 1.

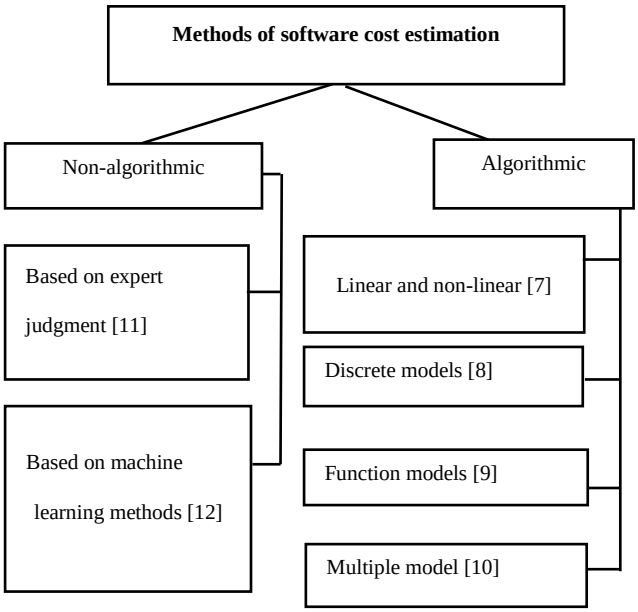

Figure 1. Categorization of cost estimation.

In general, SDEE based on algorithmic models is expressed as (1) [13].

$$EFFORT = F(x_1, x_2, \dots) \quad (1)$$

In this formula, the order x1, x2, ... expresses the features of each project. Effort estimation based on different algorithmic models is usually different from the other models. In these methods, the estimated model is formulated based on a specific algorithm. A variety of algorithmic models are shown in Fig. 1. Since these models are based on the old data and cannot consider current developments in programming languages, hardware, and software engineering, decision-making is difficult based on the results [1].

In expert judgments, there is usually no need for the old data. Expert judgment is often based on the reuse of determiner previous projects, which may not be documented perfectly. The results obtained show that 62% of the software projects in the organizations are estimated by this method. The advantage of this method is its customization for any specific organization culture, which makes it more accurate than the algorithmic methods. Also in many cases, it has been proven that it has more accuracy than the other preferred models. However, this estimation is subjective, and is based upon each logic expert. Here upon, its advantages can also be considered as its drawbacks. The estimated costs of each expert is only based on his experiences in a specific organization culture, and is not perfect in other organizations [1].

In the ML techniques, patterns of the old project data are learned, and can be used for effort prediction in new projects [1]. In SDEE, many researches have been performed by the ML approach [12]. In ML, a supervised method learns a model from the labeled training data. In the ML classification methods, labels are discrete, whereas in the regression ones, labels are continuous. Where the costs or efforts are calculated as numeric values in software projects, regression methods are studied as the ML models in SDEE researches. The ML algorithms in SDEE are divided into 6 categories [3] as case base reasoning (CBR) [12], artificial neural networks (ANNs) [14], decision trees (DT) [15], Bayesian networks (BN) [16], support vector regression (SVR) [14], and association rules (AR) [17].

Searching for useful and effective subset of features is known as the approach in the ML area to increase the learning model accuracy. Since all the ML hypotheses are potentially susceptible to the wrong, irrelevant, and redundant features [18], SCE models use a large set of features for estimation that are called cost determination. All of these features are not effective for accurate estimation. Thus in SCE, feature selection (FS) algorithms are used, which have the ability of selecting the subset of most instructive cost determination correctly, and can achieve high accuracy of ML algorithm [19].

In the ML studies, complexity and usability of classifier or regressor are dependent on the number of input features. In this area, two main methods as feature selection (FS) and feature extraction were used for feature reduction. In FS, researchers were concerned to find k features of d original features, which gave the most effective information. In feature extraction, k features were extracted from d initial features in a linear or non-linear manner [35]. Many researchers have focused on the efficiency improvement in SDEE by reducing the sample project features [34].

Different ML algorithms have been compared in [12] to estimate the cost of the software with different datasets. The effect of backward selection on each ML algorithm was studied in this work. Datasets in SDEE were divided into the within-company and cross-company categories [20-22]. In [34], the FS effect has been investigated to SDEE within-company and cross-company datasets, and it has been concluded that cost estimation with less features provides

equivalent or better accuracy than estimation with all features. In SDEE, both the filter [34] and wrapper [34] FS techniques have been used. Also in [35], a combination of filter and wrapper techniques have been developed. Studies on SCE have indicated that using the ML algorithms with dimension reduction methods can improve the accuracy.

Some of the researchers have used the isolation and connection analysis to dimension reduction in SCE [27]. Extensive research works have been conducted on finding the best subset of the cost determination, wrapper method, and climbing hills [19]. In [28], using the linear regression and wrapper FS, cost determination has been ranked based on the number of repeat times in different groups and then removing the features with lower ranks. In [29], linear regression and wrapper FSS have been implemented. The results show that a combination of pruning rows (samples) and pruning columns (features) can significantly improve the effort estimation, particularly in the small datasets.

In [30], optimum accuracy has been achieved in this area using the feature weightings and comparative methods based on euclidean distance by using filter FS. In addition, some researchers have developed genetic algorithms to achieve a suitable weight for features [31, 32].

In [19], researchers have examined the balance between the features of the old datasets to reduce cost determination, while maintaining accuracy. They have used nine known FSS methods to select the most effective features. In [33], a combined method has been provided based on the mutual information and clustering features. They have combined the supervised learning and unsupervised learning methods. In unsupervised learning, the features are clustered based on the similarity between them and the clusters using hierarchical clustering. Then in the unsupervised learning stage, the feature that is most similar to the effort feature is selected as the representative of any cluster.

In this work, a hierarchical FS approach was developed. A set of features were arranged in a descending order according to different correlation criteria in the filter methods. The start set for wrapper-based methods can be initiated by different combinations of multiple-ordered feature sets. In this study, due to the importance of the initial feature sets for convergence and accuracy in wrapper methods, a hierarchical approach was developed to achieve the advantages of both the filter and wrapper methods in SDEE. Also the evaluation criterion is an important factor that influences the effectiveness of the wrapper methods. Literature review on SDEE shows that median magnitude of relative error (MMRE) and prediction accuracy (PRED) are widely used as the evaluation criteria for the wrapper FS methods. In the second phase of the proposed evaluation function (EF) method, a fused MMRE and PRED evaluation criterion is used for improving the total accuracy results. The innovation of this paper is presented in two parts: (1) developing a hierarchical structure of the filter and wrapper methods in effective FS in SDEE, and (2) developing a fused criterion in the evaluation phase of the wrapper methods that improves semantic gap in SDEE and selects the most effective features at the same time in SCE by considering two main error rate criteria.

The remainder of this paper is structured as follows: ‘FS techniques’ section is provided in section 2. In Section 3, we describe the general framework of the proposed method. The empirical setup of implementation on a variety of datasets is described in section 4. Finally, in section 5, concluding remarks and further works are discussed in detail.

2. FS techniques

‘‘Curse of dimensionality’’ was originally discussed by Bellman in 1961. The small sample set and high dimensionality problems are two major challenges in many applications. In general, a large number of features cause the increase of complexity in data analysis and reduce the performance of learning methods such as classification, regression, and clustering. Therefore, dimensional reduction becomes an important issue for improving the efficiency. The most popular approaches in feature reduction are classified into two categories, FS and feature extraction. In FS, sample s with d features is generated from sample x with D features, where d<D. Traditional FS methods attempt to find a global optimal sub-space. It is necessary to mention that in feature extraction, the features of s are transformed into a different feature space, and thus there might be no correspondence between the two feature sets. The mathematical expressions and ideas underlying the feature extraction algorithms have been described in [34].

Heretofore, different FS methods have been proposed. These methods have been divided into three categories based on the filtering, wrapper, and embedded methods. Also these methods can be divided into two categories based on the learning dependent (wrapper, embedded) and learning independent (filter) algorithms [35]. In

the filter methods, the features are selected based on correlation of the specific criteria such as mutual information (MI) and correlation coefficients. In the wrapper methods, learning algorithms are used to determine the correlation between a subset of the features by a prediction model. In the embedded methods, the FS process and training of learning algorithms are integrated. These methods are appropriate when the feature numbers are small. One of the most common approaches in this category is learning by decision tree [35]. Since the filter and wrapper methods are used in the proposed method, these are introduced in the following section.

2.1 Filter methods

In order to check the relationship between the two features, first of all, a suitable similarity or correlation measure is required. This criterion may be considered as the function of the interaction between variables, rather than a function of their values. In this regard, correlation function may be linear or non-linear. In this function, the amount of information shared between the two variables should be considered. However, to develop this idea, quantitative information is needed. Topic of mathematics called information theory is related to correlation measurement [35]. A flowchart of the filter methods is illustrated in figure 2.

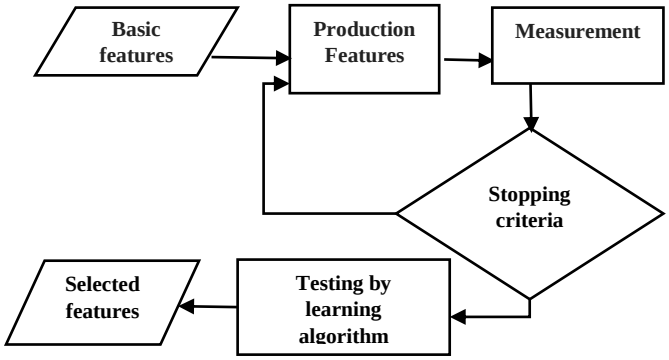

Figure 2 . Flowchart of filter methods [36].

Each filter FS method consists of three main steps: (1) production of features, (2) measurement, and (3) testing by the learning algorithm. A sub-set of features is produced in the production step. Then in the measurement step, the feature information in the current time is measured. The above two steps are performed iteratively until the results are not consistent with the assessment criteria. Afterward, the evaluation process is terminated with a threshold of measurement results. Thus maximum information must be contained in the final feature set. Test step is performed by a supervised learning algorithm.

2.2 Wrapper methods

A workflow of the wrapper method is shown in figure 3. Its process is the same as the filter methods, except that the measurement step has been replaced by a learning algorithm. This is the main reason that the wrapper methods are slow. On the other hand, the wrapper method learning algorithm can lead to better results in most cases. The process is stopped when the results obtained are worsened or the number of features reach a pre-determined threshold.

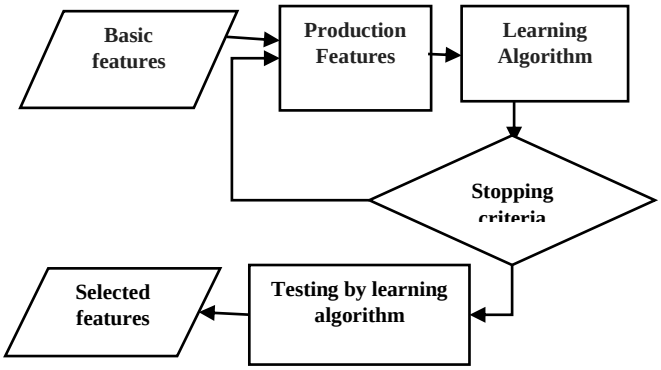

Figure 3. Flowchart of wrapper methods [36].

In regard to the point that the limited scope is most effective in the applicability of the wrapper FSS methods, the hierarchical structure of the filter and wrapper methods are used. In this approach, various combinations of filtering methods are being tested, and the most effective one is combined with the wrapper methods. Also due to the fact that the evaluating criteria in the wrapper methods impact directly on selecting effective features, in this work, hybrid criteria were utilized.

3. Proposed method

In this section, effective FS approach is presented based on utilizing a combination of both wrapper and filter. Filtering methods are faster than wrapper methods. However, the wrapper methods are more accurate than the filtering ones [37]. Thus by combining these methods, the advantages of each method can be used to eliminate the disadvantages of the other one.

In the proposed method, at first, the features are ranked based on the P filtering feature selection methods and selected TP of features that have better rank in every method as the selected features. Using the two operators AND and XOR, two final sets of proposed features are produced from the filtering methods. Then the AND set is considered as the basic one, and by using a regression algorithm, the initial accuracy is evaluated based on the fused criteria.

Furthermore, by using the two wrapper feature selection methods, the most effective features of the AND and XOR sets are selected. The AND set is considered as the input for the backward FFS method (Algorithm 2), and the XOR set is considered as input to the forward the FSS method (Algorithm 3). These two methods are repeated to increase the accuracy, and finally, the most effective features for each dataset are selected. A chart of the proposed method is shown in figure 4.

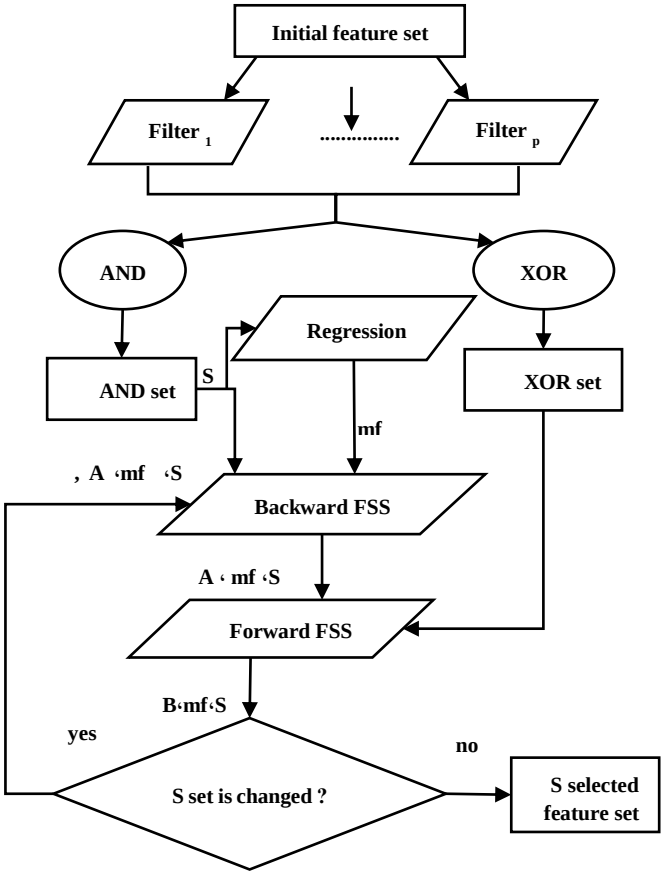

Figure 4. Flowchart of proposed method.

Various filtering methods are used in this article but in the wrapper ones, only the simple greedy forward and backward FS methods are used. Pseudo-code of the proposed method for combination of these methods is represented in Algorithm 1. In this method, using the fused function within the wrapper methods, a combination of criteria are generated for assessing the effectiveness of the selected features. This approach causes a higher reduction in the semantic gap by selecting the effective features. For this purpose, all evaluation criteria are passed to fused function. The result is a combined criterion as m that inherits all criteria measures. Two sets of A and B are constructed from the output of the filter methods. Common features of the filter methods are assigned to A, and consequently, non-common features are assigned to B. The rest of the proposed method is followed by the two wrapper methods (Algorithms 2, 3) in an iterative manner. This iteration continues until the accuracy is converged to an optimal value. The output of the proposed method is the selected feature set.

ALGORITHM 1. Hierarchical FS algorithm

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 [x1, x2… xD], where D is the number of sample features.*

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

$F=\{f_i\}_{i=1}^{P}$ *where f_i is the* 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 tp selected features by fi on X.*

end

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

$A=AND_1^P\ Sp$

$R=XOR_1^P\ Sp$

s= A

mf=regression(X,s,m), where mf is accuracy result evaluated by m.

repeat

[s A mf]= BFS-Function (X,s, A,m,mf) *backward FS*

[s R mf]= FFS-Function (X,s, B,m,mf) *forward FS*

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

Output:

s, where s is the optimum subset of original features

ALGORITHM 2. 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 [x1, x2… xD], where D is the number of sample features.*

S, *where S is the initial subset for backward FS.*

A, *where A is the 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 3. 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 [x1, x2… xD], where D is the number of sample features.*
S, *where s is the initial subset for forward FS.*
B, *where B is the additive subset for forward FS.*
M, *where m is the measurement criterion.*
Mf, *where Mf is the accuracy result of 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.

4. Empirical setup

In this section, the implementation and analysis of experimental results in different datasets are represented. First the criteria and datasets used are proposed. Then the results are presented, and finally, the results are compared and verified by the results of other researches. With the purpose of implementing the proposed methods, the FEAST tools are used, taken from [38].

4.1. Performance metrics

In this paper, in order to evaluate the accuracy of this idea in the SDEE, the proposed method implements various datasets in these fields, and the evaluation criteria of these fields are used to analyze the results. In this field, various evaluation criteria are used. The most commonly used criteria are MRE, which represents the difference between the estimated costs and actual costs, MMRE, which represents the average estimation error for the total sample (training samples and test samples), and PRED(X), which represents the percentage of samples whose magnitude of relative error 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 formula used for the criterion defined above will be followed.

$$MRE = \frac{|Actual\ Effort - Estimated\ Effort|}{Actual\ Effort} \quad (2)$$

- *Actual Effort* is the real project's effort.
- *Estimated Effort* is the estimated Effort by the algorithm.

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|Estimated(i) - Actual(i)|}{Actual(i)}\right) \quad (3)$$

- *Actual Effort* is the real project's effort.
- *Estimated Effort* is the estimated Effort by the algorithm.

$$MDMRE = Median\ (MRE) \quad (4)$$

$$PRED(x) = \frac{k}{n} \quad (5)$$

- *X* is the difference in most research works that is equal to 0.25.
- *K* is the number of samples, and the difference between their estimated cost and their actual cost is equal or less than *x*.
- *N* is the total number of tested samples.

Therefore, the higher value for PRED (0.25) results in the less error rate of the evaluated algorithm, and the estimated cost for the number of tested sample error rate is equal to or less than 0.25. Thus by picking up the features that result in lower MMRE and higher PRED, the semantic gap can be reduced in the estimation procedure.

4.2. Datasets

In this study, three popular datasets in SDEE (cocomo81, coconasa93, and Desharnais) were used. They will be briefly introduced, and their usage will be followed. Studies on SDEE with the ML algorithms using the validation methods divides the dataset into two groups, training dataset and test dataset, from which the former is used for learning algorithms. Accuracy of the test dataset is the main goal of the results of evaluating the algorithm accuracy. For this purpose, in this study, two methods titled LOOCV and 10-fold cross validation were used.

Cocomo81

This dataset consists of a variety of 63 commercial, scientific, systematic, proactive, and supportive software projects. There are 16 independent variables that are determined by the product, project computers and personal characteristics by hour per person [19]. The dataset features are listed in table 1.

Coconasa93

This dataset includes 93 examples of projects implemented using different canters of NASA, during 1971 to 1987, with 23 independent features, which consist of 16 common features with cocomo81 and 7 other independent features and one dependent feature of effort. These features are categorical. The first pre-processing was carried out in this study, transforming the categorical data to the numerical data. Other features of coconasa93 that are not present in cocomo81 are listed in table 2.

Table 1. Cocomo81 dataset features.

No	Feature	Full name
X_1	rely	required software reliability
X_2	data	data base size
X_3	cplx	process complexity
X_4	time	time constraint for cpu
X_5	stor	main memory constraint
X_6	virt	machine volatility
X_7	turn	turnaround time
X_8	acap	analysts capability
X_9	aexp	application experience
X_{10}	pcap	programmers capability
X_{11}	vexp	virtual machine experience
X_{12}	lexp	language experience
X_{13}	modp	modern programing practices
X_{14}	tool	use of software tools
X_{15}	sced	schedule constraint
X_{16}	loc	Line of code
X_{17}	Effort	Effort

Table 2. Surplus features than cocomo81 in coconasa93.

No	Feature	Full name
X_{18}	Unique id	record number real
X_{19}	project name	project name
X_{20}	cat2	category of application
X_{21}	frog	Flight or ground system?
X_{22}	center	which nasa center
X_{23}	year real	year of development
X_{24}	mode	development mode

Desharnais

The original version of this dataset contains 81 projects of generated projects by a candidate software house that have been described in 12 features. The second and third features in four samples are miss value. For this reason, this dataset has been used differently in various articles. In some papers, 4 samples have been put aside and the other 77 samples have been used [19]. Other researchers have removed the miss value columns from the set of columns [39]. In this work, both methods in these datasets were used. The features of this dataset are described in table 3.

Table 3. Original Desharnais dataset features.

No	Feature	Full name
X_1	Project	Project id
X_2	TeamExp	Team experience (measured in years)
X_3	ManagerExp	Manager's experience (measured in years).
X_4	Yearend	Year End
X_5	Length	Project's duration (measured in months).
X_6	Effort	Actual development effort (in person-hours)
X_7	Transactions	Count of basic logical transactions in the system
X_8	Entities	Number of entities in the system's data model.
X_9	Points Adjust	Adjusted Function Points count.
X_{10}	Envergure	Scale of the project.
X_{11}	PointsNonAjust	Unadjusted Function Points count
X_{12}	language	Language

4.3. Experimental result

Here are the results of various tests on the test datasets introduced in the previous section. The best results for each dataset were marked bold. Some studies have used a combination of the MMRE and PRED criteria for ranking the algorithms used in this field that are displayed with the EF symbol [40]. This method is produced by fusion function.

$$EF = \frac{PRED(0.25)}{1 + MMRE} \quad (6)$$

In this study, EF that is a combination of the criteria for FS is used. The selected features provide a higher EF. In the research works carried out in the FS field in SDEE, usually the MMRE criteria and, less often, the PRED criteria are used for FS. As mentioned, we looked for the features that provided a lower MMRE and higher PRED. Thus when the EF criteria are used, the selected features will have these two conditions. In this work, a multi-layer perceptron (MLP) neural network learning algorithm was developed for the wrapper FS. Artificial neural networks (ANNs) contain a lot of highly inter-connected processing elements called neurons. They usually operate in parallel, and are configured with a regular architecture. Each neuron is connected via a communicative link with other neurons. Each communicative link has a weight that represents the information about the input signal. Neuron calculates a sum of input weights, and if the total weight is more than a threshold, produces an output. This process continues until one (or more) output(s) is (are) produced. The estimate models can be trained using the old training data to produce the results by fine-tuning the algorithm parameter values to reduce the difference between the actual and estimated efforts [34]. The MLP neural network in this study consisted of an input layer, a hidden layer, and an output layer. The parameters of the proposed algorithm in this paper were set by the values presented in table 4.

Table 4. Values of method parameters in this work.

Parameter	Value
k	3
p	2
T_p (cocomo81)	10
T_p (coconasa93)	10
T_p (Desharnais with 81 sample)	4
T_p (Desharnais with 77 sample)	5

Based on the results of the implementation of various compounds in the Desharnais dataset with 77 cases and 12 features, it is clear that the method is effective and from different combinations; only 2 cases have reduced accuracy. Among the compounds tested, 9 different combinations achieved the highest possible accuracy. The 10-Fold cross-validation method was used to evaluate the dataset. This data was divided into 10 equal parts, one as the test data and the other 9-folds were considered as the training data. Similar to similar articles, this process were carried out for ten times and the average results were presented. In Desharnais, for the dataset consisting of 81 samples and 9 features, all combinations increased accuracy. In

fact, the accuracy of all compounds was greater than that for the conventional MLP. Among the various compounds, by combining the Mifs and Relief filter methods, the highest accuracy can be achieved. From the experimental results of the Cocomo81 dataset, it can be concluded that a combination of different FS methods in the dataset is effective and has a higher accuracy. Based on the comparison of different combinations in the filtering step, except one compound, all the combinations caused a higher accuracy. A combination of two methods, Betagamma and Relief, provided the highest accuracy. In the coconasa93 dataset, we used the LOOCV validation method to evaluate the technique. In this form, the dataset was divided for 93 times, containing 92 training and one testing samples. Finally, the average of the results of 93 times division was presented. The results obtained showed that the method was effective in the coconasa93 dataset. From 37 different compounds, 31 compounds provided an accuracy higher than the simple MLP algorithm. The best accuracy was the result of a combination of the MRMR and Cief filter methods, and the lowest accuracy was the result of a combination of the Cief and Icap methods. The results of the implementation of these methods in cocomo81 and coconasa93 are shown in table 5, and the results of its implementation in Desharnais with both approaches are presented in table 6.

According to different compounds, **Size** is known as the most effective feature, which is common in all datasets. Also the two features **Cplx** and **Tool** of COCOMO81, two features **VIRT** and **VEXP** of COCONASA93, and the **Transactions** and **Entities** features from Desharnais with 81 samples and **length, entities** and **envergure** features of Desharnais with 77 samples in all compounds were identified as excess features (less important one).

Table 5. Results of implementation on Desharnais dataset.

Dataset	**Deshar77**				**Deshar88**			
Method	**MMRE**	**MDMRE**	**PRED**	**EF**	**MMRE**	**MDMRE**	**PRED**	**EF**
No method	0.6296	0.4492	31.1688	19.1262	0.9067	0.4574	28.3951	14.8923
Jmi,MRMR	0.6613	0.3485	43.0357	25.9041	0.6211	0.3536	39.5062	24.3702
Jmi, Mifs	0.4952	0.3568	40.3571	26.9909	0.6211	0.3536	39.5062	24.3702
Jmi,Disr	0.6613	0.3485	43.0357	25.9041	0.6082	0.3479	39.5062	24.5654
Jmi,Icap	**0.4742**	**0.3320**	**41.6071**	**28.2240**	0.6082	0.3479	39.5062	24.5654
Jmi,Condred	0.6774	0.3467	42.8571	25.5499	0.6211	0.3536	39.5062	24.3702
Jmi,Betagamma	0.6774	0.3467	42.8571	25.5499	0.6211	0.3536	39.5062	24.3702
MRMR,Mifs	0.4952	0.3568	40.3571	26.9909	0.6211	0.3536	39.5062	24.3702
MRMR,Disr	0.6613	0.3485	43.0357	25.9041	0.6211	0.3536	39.5062	24.3702
MRMR,Icap	**0.4742**	**0.3320**	**41.6071**	**28.2240**	0.6211	0.3536	39.5062	24.3702
MRMR,Condred	0.6774	0.3467	42.8571	25.5499	0.6211	0.3536	39.5062	24.3702
MRMR,Betagamma	0.6774	0.3467	42.8571	25.5499	0.6211	0.3536	39.5062	24.3702
MRMR,Relief	**0.4742**	**0.3320**	**41.6071**	**28.2240**	0.6527	0.3437	40.7407	24.6516
Mifs,Condred	0.6680	0.3651	40.1786	24.0873	0.6211	0.3536	39.5062	24.3702
Mifs,Relief	0.5154	0.3643	36.4286	24.0391	**0.6182**	**0.3245**	**40.7407**	**25.1773**
Disr,Mifs	0.4952	0.3568	40.3571	26.9909	0.6211	0.3536	39.5062	24.3702
Disr,Condred	0.6774	0.3467	42.8571	25.5499	0.6211	0.3536	39.5062	24.3702
Disr,Betagamma	0.6774	0.3467	42.8571	25.5499	0.6211	0.3536	39.5062	24.3702
Cief,Jmi	**0.4742**	**0.3320**	**41.6071**	**28.2240**	0.6082	0.3479	39.5062	24.5654
Cief,MRMR	**0.4742**	**0.3320**	**41.6071**	**28.2240**	0.6211	0.3536	39.5062	24.3702
Cief,Mifs	0.5106	0.3573	36.4286	24.1148	0.6211	0.3536	39.5062	24.3702
Cief,Disr	0.4742	0.3320	41.6071	28.2240	0.6082	0.3479	39.5062	24.5654
Cief,Condred	0.5148	0.3726	42.1429	27.8204	0.6211	0.3536	39.5062	24.3702
Icap,Mifs	0.5106	0.3573	36.4286	24.1148	0.6362	0.3860	37.0370	22.6363
Icap,Disr	**0.4742**	**0.3320**	**41.6071**	**28.2240**	0.6082	0.3479	39.5062	24.5654
Icap,Cief	0.7081	0.4517	29.8214	17.4584	0.6362	0.3860	37.0370	22.6363
Icap,Betagamma	0.5148	0.3726	42.1429	27.8204	0.6211	0.3536	39.5062	24.3702
Condred,Icap	0.5148	0.3726	42.1429	27.8204	0.6211	0.3536	39.5062	24.3702
Condred,Relief	0.5148	0.3726	42.1429	27.8204	0.6211	0.3536	39.5062	24.3702
Betagamma,Mifs	0.6680	0.3651	40.1786	24.0873	0.6211	0.3536	39.5062	24.3702
Betagamma,Cief	0.5148	0.3726	42.1429	27.8204	0.6211	0.3536	39.5062	24.3702
Betagamma,Condred	0.6492	0.3451	41.7857	25.3372	0.6211	0.3536	39.5062	24.3702
Relief,Jmi	**0.4742**	**0.3320**	**41.6071**	**28.2240**	0.6211	0.3536	39.5062	24.3702
Relief,Disr	**0.4742**	**0.3320**	**41.6071**	**28.2240**	0.6082	0.3479	39.5062	24.5654
Relief,Cief	0.7098	0.4351	31.0714	18.1727	0.6527	0.3437	40.7407	24.6516
Relief,Icap	0.7098	0.4351	31.0714	18.1727	0.6527	0.3437	40.7407	24.6516
Relief,Betagamma	0.5148	0.3726	42.1429	27.8204	0.6211	0.3536	39.5062	24.3702

Table 6. Results of implementation on cocomo81 and coconasa93 datasets.

Dataset	Coconasa93				Cocomo81			
Method	**MMRE**	**MDMRE**	**PRED**	**EF**	**MMRE**	**MDMRE**	**PRED**	**EF**
No method	1.2484	0.4193	30.1075	13.3908	1.9239	0.6926	20.6349	7.0572
Jmi,MRMR	1.1433	0.3908	39.7849	18.5626	1.7703	0.7331	22.2222	8.0216
Jmi,Mifs	0.9439	0.4226	36.5591	18.8075	2.1427	0.5026	28.5714	9.0913
Jmi,Disr	1.1216	0.3711	38.7097	18.2454	1.9773	0.6803	26.9841	9.0633
Jmi,Icap	1.4547	0.4453	34.4086	14.0172	1.6392	0.5820	23.8095	9.0214
Jmi,Condred	0.9989	0.4103	35.4839	17.7519	1.7009	0.6477	22.2222	8.2277
Jmi,Betagamma	0.9989	0.4103	35.4839	17.7519	1.7703	0.7331	22.2222	8.0216
MRMR,Disr	1.1632	0.3724	38.7097	17.8948	1.9773	0.6803	26.9841	9.0633
MRMR,Cief	**1.1865**	**0.3398**	**44.0860**	**20.1627**	1.6392	0.5820	23.8095	9.0214
MRMR,Condred	1.1632	0.3724	38.7097	17.8948	1.7009	0.6477	22.2222	8.2277
MRMR,Betagamma	1.1632	0.3724	38.7097	17.8948	1.6170	0.7323	20.6349	7.8849
MRMR,Relief	1.0595	0.4062	36.5591	17.7512	1.6760	0.6917	25.3968	9.4905
Mifs,MRMR	1.0591	0.4053	37.6344	18.2768	2.1427	0.5026	28.5714	9.0913
Mifs,Disr	0.9439	0.4226	36.5591	18.8075	2.1427	0.5026	28.5714	9.0913
Mifs Icap	1.2549	0.4766	32.2581	14.3055	2.1427	0.5026	28.5714	9.0913
Mifs,Condred	0.9439	0.4226	36.5591	18.8075	1.7672	0.6637	23.8095	8.6041
Mifs,Betagamma	0.9439	0.4226	36.5591	18.8075	2.1427	0.5026	28.5714	9.0913
Mifs,Relief	0.9238	0.4657	31.1828	16.2093	1.7611	0.6973	22.2222	8.0483
Disr,Cief	1.1808	0.3773	41.9355	19.2298	1.7484	0.6544	26.9841	9.8182
Disr,Icap	0.8894	0.6938	20.4301	10.8128	1.7484	0.6544	26.9841	9.8182
Disr,Condred	1.1444	0.3560	40.8602	19.0547	1.7710	0.7567	23.8095	8.5925
Disr,Betagamma	1.1444	0.3560	40.8602	19.0547	1.9773	0.6803	26.9841	9.0633
Cief,Jmi	1.1821	0.3640	40.8602	18.7252	1.6392	0.5820	23.8095	9.0214
Cief,Disr	1.1808	0.3773	41.9355	19.2298	1.7484	0.6544	26.9841	9.8182
Cief,Icap	3.5438	0.8240	17.2043	3.7864	1.7484	0.6544	26.9841	9.8182
cief,Betagamma	1.2384	0.4104	30.1075	13.4503	2.1774	0.5798	28.5714	8.9920
Cief,Relief	3.1438	0.7860	18.2796	4.4113	1.6271	0.6293	22.2222	8.4589
Icap,MRMR	1.4198	0.4650	35.4839	14.6639	1.6392	0.5820	23.8095	9.0214
Icap,Condred	0.9534	0.6348	25.8065	13.2108	2.3809	0.6853	25.3968	7.5119
Icap,Betagamma	0.9534	0.6348	25.8065	13.2108	1.6392	0.5820	23.8095	9.0214
Icap,Relief	2.5552	0.7525	16.1290	4.5368	1.4194	0.7333	22.2222	9.1852
Condred,Cief	1.2384	0.4104	30.1075	13.4503	1.6753	0.5807	28.5714	10.6797
Condred,Betagamma	1.1691	0.3753	36.5591	16.8546	1.7009	0.6477	22.2222	8.2277
Condred,Relief	1.0188	0.4816	34.4086	17.0440	1.6606	0.6923	22.2222	8.3523
Betagamma,Disr	1.1472	0.3912	36.5591	17.0263	1.9773	0.6803	26.9841	9.0633
Betagamma,Relief	1.0188	0.4816	34.4086	17.0440	**1.6352**	**0.7146**	**28.5714**	**10.8421**
Relief,Jmi	1.1399	0.3698	39.7849	18.5921	1.7681	0.7319	20.6349	7.4545
Relief, Disr	1.1170	0.3377	39.7849	18.7933	2.4197	0.7967	22.2222	6.4983

4.4. Comparing the results with other works

In this section, in order to verify the accuracy of the proposed method in this work, the enhanced experimental results are compared with other studies [12]. Here, we should emphasize that the validation method used in this paper is similar to the method used in comparative literature. As shown in table 7, the results of this work are better than the results of comparative literature in all datasets. In evaluation, the best results of experiments on each dataset were compared with the works done by others. In [12], different learning algorithms have been implemented on different datasets, and due to the MLP usage in this study, the best results of [12] with the MLP algorithm after FS have been considered in comparison. It is necessary to repeat that the original coconasa93 dataset has 93 projects with 24 features. In [12], this dataset with 16 features has been used. In [12], "* 100" has been removed from the MDMRE formula. In other words, in their presented formula, the output has not been multiplied by 100. In order to create the conditions to compare, their results have been multiplied by 100. In table 7, a comparison is presented between the experimental results of this paper and other studies.

5. Conclusion

In this work, a hierarchical FS approach in SCE was developed. In the proposed approach, the accuracy and time complexity were improved. Using the wrapper methods, the learning algorithm must be run in each round for evaluating the effectiveness of each feature. Thus the filter methods were utilized for limiting the scope of the search into the most effective features, which reduce the number of search in the wrapper methods, and consequently, have a lower computational complexity.

The filter methods have higher speeds, while their accuracy is not acceptable. The wrapper methods have lower speeds, and due to the use of ML

algorithm, they have higher accuracy. Combination of the filter and wrapper methods resulted in an optimal performance by eliminating the weaknesses of each approach and using the advantages of the other ones. This method was evaluated on the cocomo81, coconasa93, and Desharnais datasets. The results obtained indicated the effectiveness of the method. According to different compounds, the common feature of all datasets, "size" feature, is known as the most effective one.

Table 7. Comparison between experimental results of this work and other studies.

Dataset	Paper		MMRE	MDMRE	PRED	EF
COCOMO81+	This Paper		**1.6352**	**0.7146**	**28.5714**	**10.8421**
LOOCV	[12]		-	0.79	17.5	-
Coconasa93+	This Paper		**1.1865**	**0.3398**	**44.0860**	**20.1627**
LOOCV	[12]		-	0.38.5	37.6	-
Desharnais81+	This Paper		**0.6182**	**0.3245**	**40.7407**	**25.1773**
LOOCV	[39]		0.6480	0.3708	-	-
Desharnais77+	This Paper		**0.4742**	**0.3320**	**41.6071**	**28.2240**
10-Fold	[19]	BFE	0.592	-	37.7	23.71
		FFS	0.577	-	38.2	24.22
		BSWF	0.618	-	38.6	23.85
		FSWF	0.600	-	37.3	23.31
		LSBFE	0.592	-	37.2	23.36
		LSFFS	0.596	-	36.3	22.74
		GARSON	0.606	-	44.7	27.83
		LSGA	0.607	-	35.5	22.09
		GA	0.570	-	37.6	23.94

In the future, we intend to work on other combinations of the filter and wrapper methods. In this study, we used a combination of the filter methods in the first phase. Composition of more filter methods may provide more accuracy. In this work, we used a multi-layer neural network algorithm as a learning algorithm. In the future works, we intend to implement this approach in the other learning algorithms. In this work, the EF criteria for SF in the SCR were used for the first time. This criterion consists of a combination of two important evaluation criteria used in other articles in this issue. The proposed method has a function to combine the different evaluation criteria used in this field. We are going to provide more powerful combinations of evaluation criteria using techniques such as genetic programming by fused function.

References

[1] Pandey, P. (2013). Analysis of the Techniques for Software Cost Estimation. International Journal of Software Engineering Research & Practices, vol. 3, no. 1, pp. 9-15.

[2] Wal, A. S., Gupta, V., & Kumar, R. (2012). Emerging Estimation Techniques. International Journal of Computer Applications, vol. 59, no. 8, pp. 30-34.

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

[4] Kumari, S., & Pushkar, S. (2013). Comparison and analysis of different software cost estimation methods. International Journal of Advanced Computer Science and application, vol. 4, no. 1, pp. 153-157.

[5] Keim, Y., Bhardwaj, M., Saroop, S., & Tandon, A. (2014). Software Cost Estimation Models and Techniques: A Survey. In International Journal of Engineering Research and Technology, vol. 3, no. 2, pp. 1763-1768.

[6] Zaid, A., Selamat, M. H., Ghani, A. A. A., Atan, R., & Wei, K. T. (2008). Issues in software cost estimation. International Journal of Computer Science and Network Security, vol. 8, no.11, pp. 350-356.

[7] Bailey, J. W. & Basiii, V. R. (1981). A meta-model for software development resource expenditures. 5th international conference on Software engineering, IEEE press, 1981.

[8] Wolverton, R. W. (1974). The cost of developing large-scale software. Computers. IEEE Transactions on Software Engineering, vol. 100, no. 6, pp. 615-636.

[9] Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. IEEE transactions on Software Engineering, vol. 4, no. 4, pp. 345-361.

[10] Venkatachalam, A. R. (1993). Software cost estimation using artificial neural networks. International Joint Conference on Proceedings, IEEE press, 1993.

[11] Helmer, O. (1966). Social technology (vol. 9). New York: Basic Books.

[12] Dejaeger, K., Verbeke, W., Martens, D., & Baesens, B. (2012). Data mining techniques for software effort estimation: a comparative study.

Software Engineering, IEEE Transactions on Software Engineering, vol. 38, no. 2, pp. 375-397.

[13] Bitaraf, A., yakhchi, M., Mahjobi, B. (2012), The optimal method for estimating the cost of using software-based Perceptron Neural Network Data Mining. 2th National Conference on soft Computing and Information Technology, mahshahr, Iran, 2012. (in persian)

[14] Attarzadeh, I., & Ow, S. H. (2010). Proposing a new software cost estimation model based on artificial neural networks. 2th International Conference on Computer Engineering and Technology (ICCET), IEEE press, 2010.

[15] Nassif, A. B., Azzeh, M., Capretz, L. F., & Ho, D. (2013). A comparison between decision trees and decision tree forest models for software development effort estimation. 3th International Conference on Communications and Information Technology (ICCIT), Chennai, India, 2013.

[16] Radlinski, L. (2010). 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.

[17] Bibi, S., Stamelos, I., & Angelis, L. (2008). Combining probabilistic models for explanatory productivity estimation. Information and Software Technology, vol. 50, no. 7, pp. 656-669.

[18] Kirsopp, C., Shepperd, M. J., & Hart, J. (2002). Search heuristics,case-based reasoning and software project effort prediction. The Genetic and volutionary Computation Conference, New York. Morgan Kaufmann Publishers, 2002.

[19] Papatheocharous, Efi, et al. (2012). Feature subset selection for software cost modelling and estimation. arXiv preprint arXiv:1210.1161.

[20] Alpaydin, E. (2014). Introduction to machine learning. MIT press.

[21] Chen, Z., Menzies, T., Port, D., & Boehm, B. (2005). Feature subset selection can improve software cost estimation accuracy. ACM SIGSOFT Software Engineering Notes, vol. 30, no. 4, pp. 1-6.

[22] Mendes, E., Lokan, C., Harrison, R., & Triggs, C. (2005). A replicated comparison of cross-company and within-company effort estimation models using the ISBSG database. 11th IEEE International Symposium, IEEE press, 2005.

[23] Mendes, E., & Lokan, C. (2009). Investigating the use of chronological splitting to compare software cross-company and single-company effort predictions: a replicated study. 13th Conference on Evaluation & Assessment in Software Engineering (EASE), BCS, 2009.

[24] Mendes, E., Kalinowski, M., Martins, D., Ferrucci, F., & Sarro, F. (2014). Cross-vs. Within-company cost estimation studies revisited: An extended systematic review. 18th International Conference on Evaluation and Assessment in Software Engineering, ACM, London, England, 2014.

[25] Liu, H., Wei, R., & Jiang, G. (2013). A hybrid feature selection scheme for mixed attributes data. Computational and Applied Mathematics, vol. 32, no. 1, pp. 145-161.

[26] Li, Y. F., Xie, M., & Goh, T. N. (2009). A study of mutual information based feature selection for case based reasoning in software cost estimation. Expert Systems with Applications, vol. 36, no. 3, pp. 5921-5931.

[27] Menzies, T., Port, D., Chen, Z., & Hihn, J. (2005), Specialization and extrapolation of software cost models. 20th IEEE/ACM international Conference on Automated software engineering, California, USA, 2005.

[28] Chen, Z., Menzies, T., Port, D., & Boehm, B. (2005). Finding the right data for software cost modeling. Transactions on Software Engineering, vol. 22, no. 6, pp. 38–46.

[29] Jalali, O., Menzies, T., Baker, D., and Hihn, J. (2007). Column pruning beats stratification in effort estimation. International Workshop on Predictor Models in Software Engineering, Washington, USA, 2007.

[30] Auer, M., Trendowicz, A., Graser, B., Haunschmid, E., & Biffl, S. (2006). Optimal project feature weights in analogy-based cost estimation: Improvement and limitations. IEEE Transactions on Software Engineering, vol. 32, no. 2, pp. 83–92.

[31] Huang, S. J., & Chiu, N. H. (2006). Optimization of analogy weights by genetic algorithm for software effort estimation. Information and software technology, vol. 48, no. 11, pp. 1034–1045.

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

[33] Song, Q., Ni, J., & Wang, G. (2013). 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.

[34] Chahooki, M. A. Z., & Charkari, N. M. (2014). Shape classification by manifold learning in multiple observation spaces. Information Sciences, vol. 262, pp. 46-61.

[35] Brown, G., Pocock, A., Zhao, M. J., & Luján, M. (2012). Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. The Journal of Machine Learning Research, vol. 13, no. 1, pp. 27-66.

[36] Hsu, H. H., Hsieh, C. W., & Lu, M. D. (2011). Hybrid feature selection by combining filters and wrappers. Expert Systems with Applications, vol. 38, no. 7, pp. 8144-8150.

[37] Halaku, F., Eftekhari, M., Esmaeeli, A. (2011). The combination of ridge regression and correlation coefficients for select the important features of nucleotide polymorphism database (SNP), 19th ICEE, Tehran, Iran. (in Persian)

[38] Pocock, A. C. (2012). Feature selection via joint likelihood (Doctoral dissertation, University of Manchester).

[39] Kosti, M. V, et al (2010). DD-EbA: An algorithm for determining the number of neighbors in cost estimation by analogy using distance distributions. arXiv preprint arXiv:1012.5755.

[40] Araujo, R. D. A., de Oliveira, A. L. I, & Soares, S. C. B. (2009). A morphological-rank-linear approach for software development cost estimation. 21th International Conference on Tools with Artificial Intelligence, New Jersey, USA, pp. 630-636, 2009.

نشریه هوش مصنوعی و داده کاوی

کاهش شکاف معنایی تخمین هزینه‌ی نرم‌افزار با استفاده از روش انتخاب ویژگی سلسله مراتبی

صبا بیرانوند* و محمدعلی زارع چاهوکی

دانشکده مهندسی برق و کامپیوتر، دانشگاه یزد، یزد، ایران

ارسال ۲۰۱۵/۰۶/۲۷؛ پذیرش ۲۰۱۶/۰۴/۰۶

چکیده:

مدیریت پروژه‌های نرم‌افزاری، یکی از فعالیت‌های مهم در فرآیند توسعه‌ی نرم‌افزار می‌باشد. تخمین تلاش توسعه‌ی نرم‌افزار (SDEE)، یکی از مسائل چالشی در مدیریت پروژه‌ی نرم‌افزاری محسوب می‌شود که قدمت آن به دهه‌ی ۱۹۹۰ برمی‌گردد. یک مدل SDEE زمانی مناسب است که دو عامل دقت و اطمینان را به صورت همزمان و قبل از عقد قرارداد فراهم نماید. با توجه به ماهیت غیر قطعی تخمین‌های نرم‌افزاری و به منظور افزایش دقت، محققان به تازگی بر تکنیک‌های یادگیری ماشین متمرکز شده‌اند. انتخاب موثرترین ویژگی‌ها برای رسیدن به دقتی بالا در یادگیری ماشین بسیار مهم است. در این مقاله، برای کاهش شکاف معنایی در SDEE، تکنیک انتخاب ویژگی سلسله مراتبی مبتنی بر فیلتر و بسته‌بندی به همراه یک معیار ارزیابی ترکیبی به صورت رویکردی دو مرحله‌ای ارائه شده است. در مرحله‌ی اول، دو روش فیلتر، مجموعه‌های اولیه ویژگی‌ها را برای تکنیک‌های انتخاب ویژگی بسته‌بندی فراهم می‌کنند. در مرحله‌ی دوم، معیاری ترکیبی برای ارزیابی دقت در تکنیک‌های انتخاب ویژگی بسته‌بندی ارائه شده است. نتایج آزمایشات نشان‌دهنده‌ی اعتبار و کارآمدی رویکرد پیشنهادی برای SDEE روی انواع دادگان می‌باشد.

کلمات کلیدی: تخمین تلاش توسعه‌ی نرم‌افزار، تخمین هزینه‌ی نرم‌افزار، یادگیری ماشین، انتخاب ویژگی سلسله مراتبی.