Shahid Bahonar University of Kerman
Journal of Mahani Mathematical Research
Print ISSN: 2251-7952 Online ISSN: 2645-4505

GKRR: A GRAVITATIONAL-BASED KERNEL RIDGE REGRESSION FOR SOFTWARE DEVELOPMENT EFFORT ESTIMATION

M.B. Dowlatshahi ✉, M.A. Zare-Chahooki, S. Beiranvand, and A. Hashemi

Dedicated to sincere professor Mashaallah Mashinchi

Article type: Research Article
(Received: 07 February 2022, Received in revised form: 07 July 2022)
(Accepted: 25 July 2022, Published Online: 06 August 2022)

Abstract. Software Development Effort Estimation (SDEE) can be interpreted as a set of efforts to produce a new software system. To increase the estimation accuracy, the researchers tried to provide various machine learning regressors for SDEE. Kernel Ridge Regression (KRR) has demonstrated good potentials to solve regression problems as a powerful machine learning technique. Gravitational Search Algorithm (GSA) is a metaheuristic method that seeks to find the optimal solution in complex optimization problems among a population of solutions. In this article, a hybrid GSA algorithm is presented that combines Binary-valued GSA (BGSA) and the real-valued GSA (RGSA) in order to optimize the KRR parameters and select the appropriate subset of features to enhance the estimation accuracy of SDEE. Two benchmark datasets are considered in the software projects domain for assessing the performance of the proposed method and similar methods in the literature. The experimental results on Desharnais and Albrecht datasets have confirmed that the proposed method significantly increases the accuracy of the estimation comparing some recently published methods in the literature of SDEE.

Keywords: Parameter Optimization, Software Development Effort Estimation, Gravitational Search Algorithm, Kernel Ridge Regression, Feature Selection.
2020 MSC: 62J07.

1. Introduction

Successful project finalizing within time is an essential mission for software industries. In recent decades, about 30 percent of software projects have failed. Incorrect software estimates can be considered as one of the most crucial reasons for these failures. Cost overruns and schedule delays are due to incorrect understanding of software efforts. A software project leads to failure because of these factors [13], [14].

✉ dowlatshahi.mb@lu.ac.ir, ORCID: 0000-0002-4862-0626
DOI: 10.22103/jmmr.2022.18988.1202
Publisher: Shahid Bahonar University of Kerman
How to cite: M.B. Dowlatshahi, M.A. Zare-Chahooki, S.Beiranvand, A.Hashemi, *GKRR: A gravitational-based kernel ridge regression for software development effort estimation*, J. Mahani Math. Res. 2022; 11(3): 147-174.

On the other hand, the competitiveness and reputation of a company can be affected by such failures. Also, overestimating software project effort may lead to missed opportunities to fund other projects due to ineffective use of resources and loss of resources project tenders. Hence, the construction of accurate methods for Software Development Effort Estimation (SDEE) represents an uninterrupted activity of researchers and software designers. In the literature, several SDEE techniques exist which can be partitioned into three categories [58], [38], [42]: (1) Expert judgment, where the process of estimating the new software projects effort is conducted by a project estimator based on her or his domain knowledge; (2) Algorithmic models are known as the most popular category in SDEE techniques [11] and consist COCOMO [9], SLIM [53] and SEER-SEM [26]; (3) Machine learning which recently is being used instead of algorithmic models and consist of Artificial Neural Networks (ANNs) [24], Decision Tree (DT) [12], Support Vector Machine (SVM) [51], etc. Machine learning (ML) models are very effective in the SDEE field [8], [59].

This process of SDEE is conducted by machine learning techniques based on historical datasets. Thus, the accuracy of the estimation is depended on these datasets, which include a set of features. Any dataset may include irrelevant features that negatively affect the performance of the estimation model. Therefore, providing an effective feature selection method can increase the accuracy of the estimation model. Therefore, one of our approaches in this paper is to provide a feature selection method with KRR. For this purpose, a binary version of the famous GSA algorithm is considered. We use a wrapper feature selection approach to find the best subset of features. On the other hand, the KRR method is a parametric method whose performance depends on the value of these parameters. Therefore, finding the best values of parameters for regression can also increase its accuracy. We have used the real-valued version of the GSA algorithm to do this. Therefore, the method we consider in this paper is a combination of two binary and real-valued versions of GSA with real-KRR. Given the power of the GSA algorithm, we expect to provide the desired performance.

Based on our knowledge, no experimental study has been performed so far on the KRR prediction system and the hybrid GSA and simultaneously finds model parameters and an optimal feature subset. This article proposes a hybrid GSA that aggregates the binary-valued GSA (BGSA) and the real-valued GSA (RGSA) to optimize the parameters of KRR and choose the appropriate subset of features to improve the estimation accuracy of SDEE. To show that, a comparison has presented in Table 1 between the proposed method and related works in the literature based on the existence of feature selection and parameter selection and the base learners in each method.

The organization of this article is presented as follows. In Section 2 the related works are presented, and in Section 3, we present an overview of kernel-based RR, RGSA and BGSA as preliminaries. Section 4 describes the proposed algorithm. Section 5 includes the experimental results that the proposed

TABLE 1. Comparison of the proposed method and related works

Method or Reference	*BaseLearner*	*FeatureSelection*	*ParameterSelection*
Proposed method	RBF and Wevelet	Binary GSA	Continuous GSA
Oliveira et al. [48]	SVR, MLP,Tree models	Gentic Algorithm	Gentic Algorithm
Elish and Helmi [23]	SVR, MLP,ANFIS	-	-
Dolado [16]	Genetic Programming	-	-
Wen et al. [66]	Genetic Programming	-	-
Braga et al. [10]	SVR	Gentic Algorithm	Gentic Algorithm
Putnam [53]	SVR, RBF	-	-
Wani and Quadri [65]	ANN	-	Bee Colony Algorithm
Azzeh et al. [7]	SVM	Fuzzy C-Means	-

method performance is assessed. At last, a conclusion is presented in Section 6.

2. Related Works

ML techniques conduct the estimating effort procedure for new software projects by generating a regression model based on the information from past software projects. Among the approaches proposed for training ANNs, Ridge Regression (RR), or equivalently Extreme Learning Machine (ELM), the method proposed by Huang and Siew [40] presents a good generalization performance to train Single-hidden Layer Feedforward Neural Networks (SLFNs) and also performs in a fast speed. As a type of ANNs, the SLFNs play a crucial role in classification and regression applications because they can directly approximate nonlinear mappings by input data. For SLFNs, the RR algorithm has a higher learning speed comparing the gradient-based learning algorithms such as Back-Propagation (BP) learning algorithm based on experience. In addition, many problems faced by gradient-based learning algorithms are avoided by RR, including local minima, learning rate, stopping criteria, and learning epochs [40]. Kernel Ridge Regression (KRR), or equivalently Kernel-based ELM (KELM), is one of the best implementations of RR introduced by [5], [17]. Selecting a kernel function and its specific parameters is one of the problems user's faces using KRR. Since it dramatically impacts the generalization of the algorithm, it is considered one of the critical steps in solving a problem using KRR.

Based on the results obtained by [6], we can observe Feature Subset Selection (FSS) impact in enhancing the machine learning methods performance for SDEE. FSS aims to choose a representative feature subset to attain the most information and description of the data [30], [33]. From a machine learning perspective, irrelevant features in a system lead to poor generalization for new data. Unfortunately, the assessment of all the subsets of features becomes an NP-hard problem [3]. Hence an approximate algorithm should be used to remove redundant data with tractable computations. The FSS techniques can be considered as two filter and wrapper methods [64], [69]. Filter-based methods

assess the features as a pre-processing procedure where a subset of features is chosen before initiating the learning process. The wrapper methods rank the features during the learning process and treat the learning algorithm like an objective function to find a feature subset with the highest accuracy. We can classify the wrapper methods in heuristic search algorithms and sequential selection algorithms [34]. An empty set (complete set) is first initialized to add features (remove features) in sequential selection algorithms to reach the maximum objective function. The heuristic search algorithms examine various subsets of features until an objective function is satisfied. Generating solutions based on the optimization problem or searching around in a search space are two common techniques that generate different subsets of features in heuristic search algorithms. Various methods can be included in the class of heuristic search algorithms like Tabu Search (TS) [70], Ant Colony Optimization (ACO) [50], Simulated Annealing (SA) [2], Genetic Algorithms (GA) [19], [10], [36] Particle Swarm Optimization (PSO) [18], [44], Greedy Randomized Adaptive Search Procedure (GRASP) [61], Iterated Local Search (ILS) [35], Variable Neighborhood Search (VNS) [29], , and Gravitational Search Algorithm (GSA) [20], [32]. In previous literature, some metaheuristic includes GSA, PSO, GA, and SA, were used to optimize the SVMs parameters and select the best input feature subset [43], [60], [63], [68]. Also, a PSO is employed to optimize the KRR parameters and choose the best input feature subset [4]. Genetic Programming as another method to SDEE is also investigated. In reference [66], Genetic programming and its adaptation have been studied in the field of SDEE.

In reference [62], various ANN reviews are presented for SDEE, and it is shown that using ANN in software effort prediction is better than traditional methods and more exact. Researchers in [65] proposed an artificial bee colony algorithm and functional link ANN (FLANN to deliver the most accurate SDEE. They showed that the proposed method has fast learning ability, and in multilayer neural networks without any hidden layer, FLANN can decrease the computational complexity. Researchers in [28] compared neural network systems with the COCOMO model.

In [67], the feature selection impact in improving the performance of SVR models in SDEE is shown. This paper proposed two wrapper feature selection algorithms to pre-process eight well-known datasets and estimated the cost of software development. Ricardo et al. Proposed multilayer dilation-erosion-linear perceptron) MDELP) Furthermore, compared with different techniques using different datasets [1]. In SDEE, like the other estimation methods, outliers are inevitable. Some outliers are experimentally added to datasets to evaluate estimation accuracy according to the existence of these outliers to clarify the efficacy of outliers on SDEE [49].

3. Preliminaries

3.1. **Kernel Ridge Regression.** Kernel Ridge Regression (KRR) is an effective method to prevail over the challenging issues confronted by Back-Propagation (BP) learning algorithms. RR was inspired by biological learning and was first proposed for the SLFNs and then enlarged to the generalized SLFNs [40], [41]. The RR output for input x, from the neural network structure perspective, is:

$$f_L(x) = \sum_{i=1}^{L} \beta_i h_i(x) = h(x)\beta, \tag{1}$$

where $\beta = [\beta_1, ..., \beta_L]^2$ refers to the weights of output between the L hidden layer nodes to the output node, and indicates the hidden layer output for input x, and $h_i(x)$ refers to the output of $i-$th hidden node. Consider that $h(x)$ is for transitioning data to the KRR feature space (L-dimensional) from the input space (d-dimensional). In the basic RR [48], the initial values for the parameters of the hidden node are considered randomly and remain fixed. To train an SLFN with basic RR, we should act simply the same as detecting a Least-Squares (LS) solution $\hat{\beta}$ of the linear system $H\beta = Y$:

$$\|H\hat{\beta} - Y\| = \min_{\beta} \|H\beta - Y\|. \tag{2}$$

$$H = \begin{bmatrix} h(x_1) \\ . \\ . \\ . \\ h(x_n) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \cdots & h_L(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_N) & \cdots & h_L(x_N) \end{bmatrix}. \tag{3}$$

$$H = \begin{bmatrix} y_1 \\ . \\ . \\ . \\ y_N \end{bmatrix}. \tag{4}$$

where H refers to the output matrix of the hidden layer, $x_i : h(x_i)$ indicates the hidden layer feature mapping for the $i-$th input and it is the i-th row of matrix H, shows the output of $i-$th hidden node for inputs $x_1; x_2; ...; x_N$ is the ith column of matrix H, and Y refers to the training data-target matrix.

For the linear system in equation 2 , the smallest norm LS solution can be obtained by [48]:

$$\hat{\beta} = H^{\dagger}Y, \tag{5}$$

where $H^{\dagger}$ refers to the Moore–Penrose generalized inverse of matrix H [55], the orthogonal projection is effective in RR [28]:$H^{\dagger} = (H^T H)^{-1} H^T$ if $H^T H$ is nonsingular or $H^{\dagger} = (H^T H)^{-1} H^T$ if HH^T is nonsingular. For achieving better generalization performance, based on the ridge regression concept [31] in the

output weight β computation, a positive quantity $\frac{1}{\lambda}$ is added to the diagonal of H^TH or HH^T:

$$\beta = H^T(\frac{1}{\lambda} + HH^T)^{-1}Y, \tag{6}$$

Alternatively, we can have:

$$\beta = (\frac{1}{\lambda} + H^TH)^{-1}H^TY, \tag{7}$$

where I refers to an identity matrix, the template of the basic RR algorithm is outlined in Figure 1. The corresponding output function of basic RR of an unseen input vector x is calculated by equation 1.

Algorithm (1): Template of basic RR algorithm.

Inputs: A test set $\mathbf{Xte} = \{(x_i, y_i) \mid x_i \in R^d, y_i \in R\}_{i=1}^{M}$, a training set $\mathbf{Xtr} = \{(x_i, y_i) \mid x_i \in R^d, y_i \in R\}_{i=1}^{N}$, hidden node number L, and a positive value λ.

For each hidden node i **DO**

Randomly generate the parameters of hidden node i;

Endfor

Obtain matrix **H** as the hidden layer output;

Obtain the vector of output weight: $\boldsymbol{\beta} = \mathbf{H}^T\left(\frac{\mathbf{I}}{\lambda} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{Y}$ or $\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{\lambda} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{Y}$;

For each test instance, $x \in$ Xte **DO**

$f(x) = \mathbf{h}(x)\boldsymbol{\beta}$;

Endfor

Error = the error of the regressor in the estimation of instances of the test set **Xte**;

Output: Error.

FIGURE 1. Template of basic RR algorithm.

Huang et al. [14] performed research on the Kernel Ridge Regression (KRR), which achieves equivalent or better generalization performance than basic RR

and faster than SVM. A kernel matrix is specified as the following equation if the users do not know the hidden layer feature mapping ($h(x)$).

$$\Omega_{ELM} = HH^T : \Omega_{ELM_{i,j}} = h(x_i).h(x_j) = K(x_i, x_j), \tag{8}$$

Thus the output function of KRR is simplified as:

$$f(x) = h(x)H^T(\frac{I}{\lambda} + HH^T)^{-1}Y = \begin{bmatrix} K(x, x_1) \\ . \\ . \\ . \\ K(x, x_N) \end{bmatrix}^{-1} (\frac{I}{\lambda} + \Omega_{ELM})^{-1}Y, \tag{9}$$

Therefore, $h(x)$ is unknown, and its corresponding kernel $K(u, v)$ can be used (e.g., $K(u, v) = exp(-y||u - v||^2)$. Also, no need to specify the number of hidden nodes (L). The KRR algorithm is summarized as algorithm (2) in Figure 2. Many kernel functions satisfy the Mercer condition, like polynomial kernel,

Algorithm (2): Template of kernel-based RR (KRR) algorithm.

Inputs: a test set $\mathbf{Xte} = \{(x_i, y_i) \mid x_i \in R^d, y_i \in R\}_{i=1}^{M}$, A training set $\mathbf{Xtr} = \{(x_i, y_i) \mid x_i \in R^d, y_i \in R\}_{i=1}^{N}$, and a positive value λ.

For each test instance, x∈Xte **DO**

$$f(x) = \begin{bmatrix} K(x, x_1) \\ . \\ . \\ . \\ K(x, x_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{\lambda} + \Omega_{ELM}\right)^{-1} \mathbf{Y};$$

Endfor

Error = the error of regressor in the estimation of instances of the test set **Xte**;

Output: Error.

FIGURE 2. Template of kernel-based RR (KRR) algorithm.

linear kernel, exponential kernel, and Gaussian kernel. In this article, we have used two typical kernel functions for performance analysis, and the chosen kernel functions are as follows:

(1) RBF kernel: $K(u, v) = exp(\frac{||u-v||^2}{\sigma^2})$, where σ indicates the input parameter of the RBF kernel.

(2) Wavelet kernel: $K(u,v) = cos(a\frac{||u-v||}{b})exp(-\frac{||u-v||^2}{c})$, where a, b, and c are the input parameters of wavelet kernel.

3.2. **Gravitational Search Algorithm.** The known population-based metaheuristic Gravitational Search Algorithm (GSA) is designed to solve optimization problems for continuous data [56]. The exploitation and exploration capabilities in GSA should be adjusted the same as most metaheuristics. The original GSA can only deal with optimization problems in which the solution elements are continuous real numbers. A Binary GSA (BGSA) was developed by Rashedi et al. [57] for solving optimization problems in binary-valued data. Also, for solving discrete and combinatorial optimization problems, a Discrete GSA (DGSA) was suggested by Dowlatshahi et al. [21], which works according the definition of a Path Re-linking (PR) procedure. This approach replaces the classic procedure in GSA, where each agent tends to the position of other agents. In the following subsections, the continuous and binary versions of GSA will be discussed.

3.3. **Real-valued gravitational search algorithm.** The real-valued extension of the GSA (RGSA) was first proposed for dealing with optimization problems in continuous data. In RGSA, we should first determine a set of agents in the $D-$dimensional solution space for finding the optimum solution. For each problem, the position of agents can be considered as a candidate solution and shown by the vector X_i. Higher quality agents get more excellent mass value since a heavy agent is assigned by great gravitation intensity and larger effective gravitation radius according to the Newtonian laws of gravity and motion. Based on RGSA, agents successively adjust their position X_i to get closer to the K best agents of the population.

In RGSA, suppose s searcher agents in a real-valued $D-$dimensional space. In this case, the position of the ith agent of the population determines based on the following formula [56]:

$$X_i = (x_i^1, ..., x_i^k, ..., x_i^D); i = 1, 2, ..., s, \tag{10}$$

where refers to the position of agent i in the dimension k. According to Rashedi et al. [56], the calculation of the ith agent mass value is done after computing the current population fitness based on the following formula:

$$q_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)}, \tag{11}$$

$$M_i(t) = \frac{q_i(t)}{\sum_{j=1}^{s} q_j(t)}, \tag{12}$$

where $fit_i(t)$ and $M_i(t)$ respectively show the agent i fitness value and its mass value at time t. In the following equations, $worst(t)$ and $best(t)$ are specified for a minimization-based problem [56]:

$$best(t) = Min_{j\in\{i=1,2,...,s\}} fit_j(t), \tag{13}$$

$$worst(t) = Max_{j\in\{i=1,2,\ldots,s\}} fit_j(t), \tag{14}$$

The acceleration of ith agent is computing based on the following equation according to the law of motion [56]:

$$a_i^k(t) = G(t) \sum_{j\in Kbest, j\neq i} rand_j \frac{M_j(t)}{R_{ij}(t)+\epsilon}(x_j^k(t) - x_i^k(t)), \tag{15}$$

where:

(1) $rand_j$ refers to a random value in ranges $[0, 1]$,
(2) $R_{ij}(t)$ demonstrate the Euclidean distance between agent i and agent j in a $D-$dimensional space,
(3) ϵ is a minimal value that prevents dividing by zero error,
(4) Kbest indicates top K agents based on the fitness values that K is time-dependent. At the beginning of the algorithm, it is initialized to Kinitial value, decreasing its value with time.
(5) $G_{initial}$ shows the primary value of the gravitational coefficient ($G(t)$). Based on the following equation, $G(t)$ is decreased by time to achieve a final value, G_{end}:

$$G(t) = G(G_{initial}, G_{end}, t), \tag{16}$$

A fraction of agent i's current velocity will be added to its acceleration to compute the subsequent velocity of agent i as follows [56]:

$$v_i^k(t+1) = rand \times v_i^k(t) + a_i^k(t) = rand \times v_i^k(t) + G(t)\sum_{j\in Kbest, j\neq i} rand_j \frac{M_j(t)}{R_{ij}(t)+\epsilon}(x_j^k(t) - x_i^k(t)) \tag{17}$$

where rand refers to a uniformly distributed random value in ranges $[0, 1]$. Furthermore, for computing the next position of the agent i , the following equation is used [56]:

$$x_i^k(t+1) = x_i^k(t) + v_i^k(t+1), \tag{18}$$

The procedure of the RGSA is presented in Figure 3. Parameters G and K in RGSA are used to balance the exploitation and exploration ability.

3.4. **Binary-valued gravitational search algorithm.** If we assume a binary search space as a hypercube, agents can only move to farther and nearer corners by transporting various numbers of bits. Several main concepts of the position and velocity updating procedure should be rectified. In the RGSA, each agent can move around the search space employing position vectors in the real continuous domain. As a result, the position updating concept can be performed for agents by augmenting velocities to positions by Equation 18. In a binary space, the concept of position updating is different. Since there are only two numbers (0 and 1) in binary problems, the position updating procedure cannot be implemented using Equation 18. Thus, a procedure is needed to employ velocities for altering the positions of agents from 1 to 0 or vice versa. In other words, a devised link is needed between position and velocity for rectifying the updating of positions.

Algorithm (3): RGSA.

Initialize the population;

Assess the fitness value for all agents;

Compute the mass value of all agents;

While stopping measure is not reached, **Do**

Update *Kbest*, *K*, and G;

Compute agent *i* acceleration according to Eq. (15);

Compute agent *i* velocity based on Eq. (17);

Update agent *i* position based Eq. (18);

Assess agent *i* fitness;

Compute agent *i* mass value;

End

Output: The Best discovered solution.

FIGURE 3. The procedure of the RGSA algorithm.

The position updating in binary spaces indicates exchanging between 0 and 1 values according to the velocities of agents. We want to know how to use the velocity concept in real space to update the positions in a binary space. Based on reference [48], we can consider the probability of each agent's velocity for changing the position of every agent. A transfer function is needed to update agent's positions by mapping velocity values to probability values. Hence, the BGSA and RGSA are differ based on two various components: (1) a new transfer function to transform all real velocity values to probability values in ranges [0,1], and (2) a different method for position updating using position vectors to update using the probability of their velocities. Rashedi et al. [57] proposed a function $S(v_i^k)$ to transform the velocity v_i^k into a probability value (Equation19). Note that the output value of the function $S(v_i^k)$ must be bounded in $[0, 1]$.

$$S(v_i^k(t)) = |tanh(v_i^k(t))| \tag{19}$$

The update of position vectors can be done based on the probability of their velocities after mapping velocities to probability values, as follows [57]:

$$x_i^k(t+1) = \begin{cases} complement(x_i^k(t)) & if\ rand < \ S(v_i^k(t+1)), \\ x_i^k(t) & if\ rand \geq \ S(v_i^k(t+1)) \end{cases} \tag{20}$$

where complement $(0) = 1$ and complement $(1) = 0$. Note that a fit convergence of the algorithm is achieved by limiting v_i^k into a good bound ($|v_i^k < \ v_{max}$ where a good value for v_{max} is 6).

4. **The proposed algorithm**

This article developed a hybrid GSA method, GKRR, for simultaneous feature selection and parameter optimization in the KRR. Figure 4 presents the GKRR hybrid system flowchart. The general steps of the proposed method are described below based on the flowchart presented in Figure 4. The proposed model is formed on historical datasets based on the presented flowchart. These datasets include features with values in different intervals. Thus, all features are normalized in a specific interval to create equal conditions and compare features. In order to build a learning model and evaluate its accuracy, the database is divided into training and test sets. The training part is given to the algorithm to build the desired model, and the test part is used to check the performance of the model. Like other population-based meta-heuristic algorithms, an initial population consisting of different solutions is created first in GSA. This population includes the feature selection section and parameter values for each solution. The feature selection section is selected and updated by the BGSA and the parameter selection section by the RGSA. In each iteration of the algorithm, the solutions are evaluated by evaluation criteria. If the termination condition is not met, the GSA is applied to the population again and creates a new population. This process continues until we reach the termination condition, and finally, the best solution is selected. The best solution includes the best subset of features and parameter values.

The following subsections show the design of the GKRR components.

4.1. **Motivation.** The SDEE process is one of the crucial tasks in software engineering. Making a proper estimation can be the key to the success of a project. In recent years, many researchers have tried to provide methods for SDEE. On the other hand, machine learning methods have proven to have a high ability to solve estimation problems. One well-known method that has performed well in SDEE is KRR. This estimation process is based on historical datasets. As a result, the performance of the model and the accuracy of the estimation depend on these datasets, which include a set of features. Any dataset can contain unrelated or redundant features that negatively affect the performance of the estimation model. Therefore, providing an effective feature selection method can increase the accuracy of the estimation model.

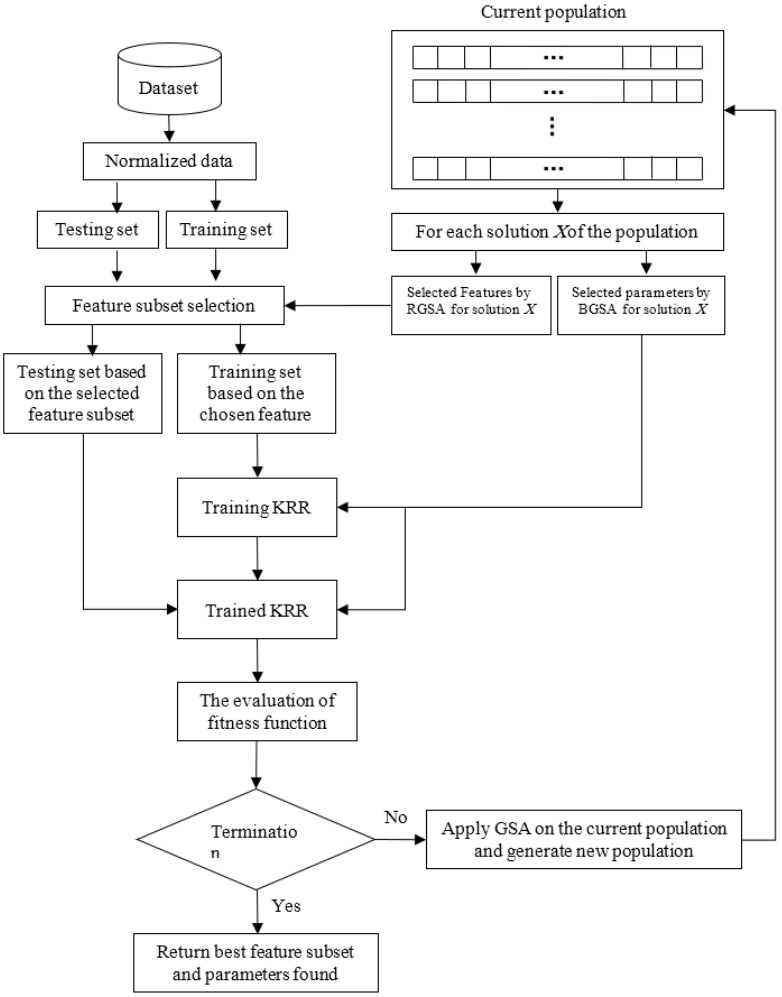

FIGURE 4. The proposed GKRR architecture

Therefore, it removes related features and noise from the data before performing the learning process using a feature selection method.

Therefore, one of our approaches in this paper is to provide a feature selection method with KRR. For this purpose, a binary version of the famous GSA algorithm is considered. We use a wrapper feature selection approach to find the best subset of features. On the other hand, the KRR method is a parametric method whose performance depends on the value of these parameters. Therefore, finding the best values of parameters for regression can also increase its accuracy. We have used the real-valued version of the GSA algorithm to

do this. Therefore, the method we consider in this paper is a combination of two binary and real-valued versions of GSA with real-KRR. Given the power of the GSA algorithm, we expect to provide the desired performance. In this algorithm, selecting the feature and parameter is done simultaneously to spend less time. So far, no similar method has used the J algorithm for this purpose.

4.2. **Solution representation.** Representation is a necessary part of each metaheuristic to encode a solution to a problem. The GKRR algorithm is represented by two separate parts: the parameter part and the feature part. The feature part includes a binary array of size d (where d refers to the number of dataset features). In other words, the feature part will be encoded by an array of d binary variables where the jth decision variable indicates the absence or presence of the jth feature. The parameter part is a real-valued array consists of the parameters of the KRR that will be optimized. Figure 5 shows a representation used by the proposed algorithm for solution $X = (1, 0, 1, 0, 0, 1.2, 2.5, 4.4)$ of a prediction problem with five features and a learning algorithm with three parameters. Note that the parameter part length of the solution representation is variable for different kernel functions because the number of parameters of kernel functions is not the same.

FIGURE 5. A two-part candidate solution in the GKRR

4.3. **Fitness evaluation.** To guide a metaheuristic towards "good" search space solutions, we need to assess the solutions' quality. Thus, a fitness evaluation function is used by metaheuristics which considers a numeric value that shows its quality with every solution of the search space. A practical fitness evaluation function should better evaluate solutions nearer to the optimal solution than farther ones. The problem solver chooses the fitness evaluation function for a given problem, which is directly concerned with the characteristics of that problem. In SDEE tasks, several metrics in the literature are available to assess the performance of prediction models. This paper employs three popular measurements are in the literature [10]: The percentage of estimates within 25 percent of the actual values (Pred(25)), Mean Magnitude of Relative Error (MMRE), and Evaluation Function (EF). The definition of each of which these measurements is as follows:

(1) Mean Magnitude of Relative Error (MMRE) of KRR on the test set $Xte = \{(x_i, t_i)|x_i \in R^d, t_i \in R\}_{i=1}^{k}$

$$MMRE = \frac{1}{M}\sum_{i=1}^{M} MRE_i \tag{21}$$

$$MMRE_i = \frac{|t_i - \hat{t}_i|}{t_i} \tag{22}$$

where M is the number of test instances, t_i is the actual target value of project i, and $\hat{t}_i$ is the estimated value of project i by KRR. Note that a lower MMRE means a more accurate estimate.

(2) $Pred(25)$ is the percentage of estimates within 25 percent of the actual values:

$$Pred(25) = \frac{100}{M}\sum_{i=1}^{M} w(MRE_i) \tag{23}$$

$$w(MRE_i) = \begin{cases} 1 & if\ MRE_i \leq 0.25 \\ 0 & if\ rand > 0.25 \end{cases} \tag{24}$$

Note that a higher $Pred(25)$ shows a more precise estimate.

(3) EF is an aggregation of MMRE and $Pred(25)$ and is defined as:

$$EF = \frac{Pred(25)}{1 + MMRE} \tag{25}$$

Note that a higher EF indicates a more accurate estimate.

In this paper, EF refers to the criterion employs for designing a fitness function for the SDEE problem. Note that the fitness function should be designed such that a solution with low MMRE and high $Pred(25)$ produces a high fitness value. Using EF measurement, this multi-criteria problem is solved by a single fitness function that aggregates the two objectives into one.

4.4. **Updating equations.** As mentioned before, this paper tries to propose a hybrid GSA to simultaneously choose the appropriate subset of features and optimize the parameters of KRR. The feature part and parameter part constitute uses the proposed algorithm representation. In each iteration, the updating of variables of the feature part is done by BGSA, and RGSA does an update of variables of the parameter part.

5. Experimental results

This section presents the assessment of the performance of the proposed method in estimating software development effort on two benchmark datasets, Desharnais [15] and Albrecht. Many articles used these datasets to assess the results of novel software effort prediction methods. The GKRR algorithm is

compared with different approaches for estimating software development effort to show the effectiveness of our method. The GKRR algorithm was implemented run on a PC with an Intel 2.53 GHz CPU and in Matlab. The GKRR parameters are adjusted as follows: 50 is agents number, the initial value of $K_{initial}$, $G_{initial}$, and G_{end} are 40, 1, and 0.01, respectively. Also, the iterations number is set to 250. Further, two linear functions are employed to decrease the value of parameters K and G by time. These parameter values are experimentally achieved.

5.1. **Datasets description.** The quality of used datasets in machine learning methods imposes on the accuracy of these methods. In SCE, 13 public datasets made available that at [46] have been studied in terms of quality. Loss of timeliness in datasets is one of the major challenges in data quality. Albrecht and Desharnais are two public datasets used more than others [46]. We have used two well-used and well-known datasets by the software engineering community in this paper. Dataset's descriptions, like the project number in the dataset, the features number, and the datasets source, are presented in Table 2.

TABLE 2. Main characteristics of two used benchmark datasets.

Dataset	number of projects	number of features	Source
Desharnais	81	11	[49]
Albrecht	24	7	[50]
Maxwell	62	27	[22]
kitchenham	145	10	[39]

5.2. **Albrecht dataset.** The Albrecht dataset contains 24 software projects were developed by using third generation languages such as COBOL, PL1, etc. 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. In Table 4, the experimental results of GKRR are given.

The obtained results show that the Gravitational-based KRR with Wavelet Kernel method provides better performance compraing the Gravitational-based KRR with RBF Kernel method in this dataset. This is demonstrated is Table 3 and Fig 6.

TABLE 3. Generated result in Albrecht dataset

Method	MMRE	Pred(25)	EF
Gravitational-based KRR with RBF Kernel	0.7322	62.50	36.07
Gravitational-based KRR with Wavelet Kernel	**0.6223**	**72.83**	**43.61**

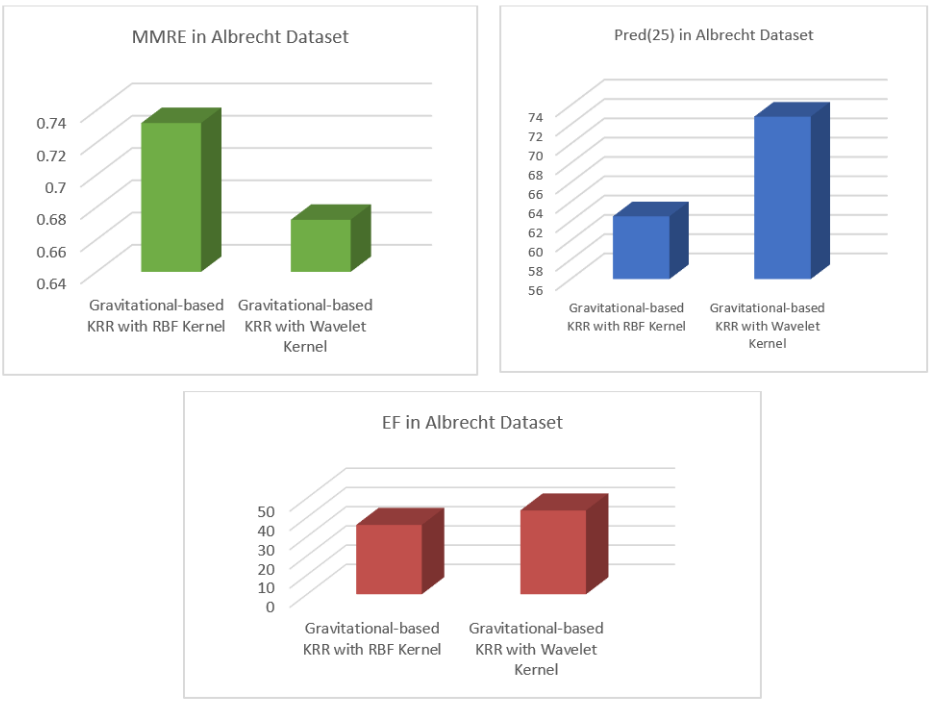

FIGURE 6. Comparison of the results obtained on Albrecht dataset

5.3. **Desharnais dataset.** This dataset encompasses projects produced during the years 1981 to 1988 in a software house. The original version of this dataset contains 81 project samples described by 12 attributes. But among all samples of the dataset, 4 samples have missing 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.

The obtained results show that the Gravitational-based KRR with Wavelet Kernel method provides better performance comparing the Gravitational-based KRR with RBF Kernel method in this dataset. This is demonstrated is Table 4 and Fig 7.

5.4. **Maxwell dataset.** It includes 62 software projects from the largest global banks in Finland, which have 26 independent variables that are determined by different software features, such as application type and size. Its dependent variable is the software development effort determined by the number of hours

TABLE 4. Generated result in Desharnais dataset

Method	MMRE	Pred(25)	EF
Gravitational-based KRR with RBF Kernel	0.2489	75.82	60.71
Gravitational-based KRR with Wavelet Kernel	**0.2145**	**78.24**	**64.42**

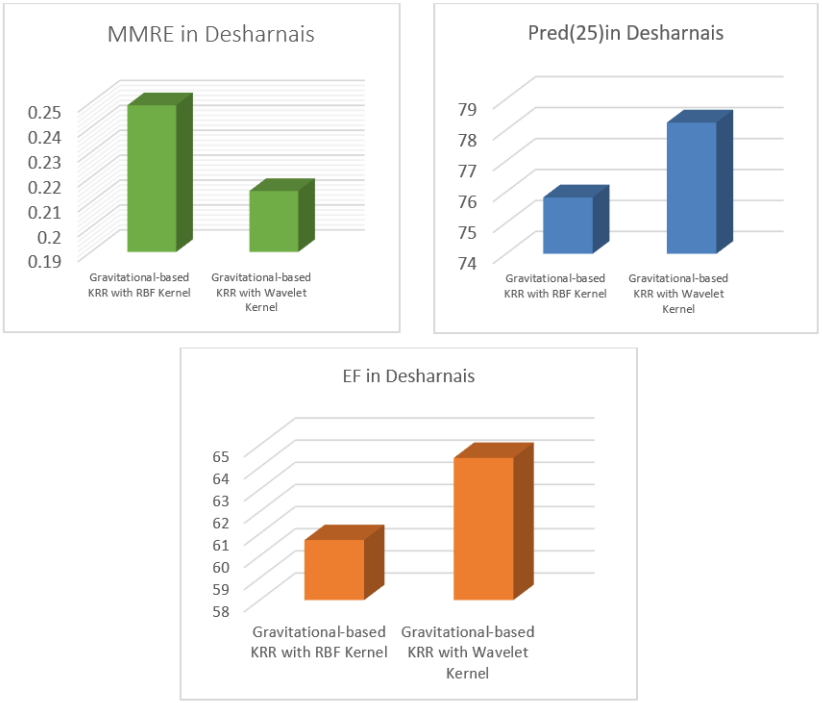

FIGURE 7. Comparison of the results obtained on Desharnais dataset

worked by the software supplier, from technical specifications to delivery time [22].

The results show that the Gravitational-based KRR with RBF Kernel performs better than the Gravitational-based KRR with Wavelet Kernel regarding the MMRE criterion. However, regarding the Pred(25) and EF criteria, the Gravitational-based KRR with Wavelet Kernel has been better than the Gravitational-based KRR with RBF Kernel. This is demonstrated is Table 5 and Fig 8.

5.5. **kitchenham dataset.** This dataset contains 145 examples and 10 attributes including project client code $\{1, 2, 3, 4, 5, 6\}$, project type $\{A, C, D, P, Pr, U\}$,

TABLE 5. Generated result in Maxwell dataset

Method	MMRE	Pred(25)	EF
Gravitational-based KRR with RBF Kernel	**0.3351**	43.29	32.42
Gravitational-based KRR with Wavelet Kernel	0.3493	**44.86**	**33.25**

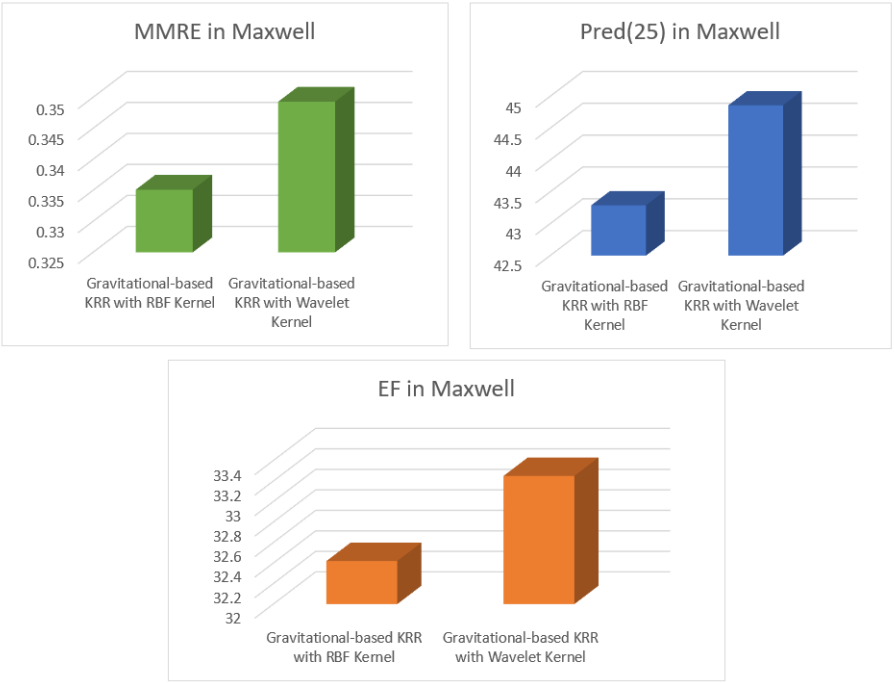

FIGURE 8. Comparison of the results obtained on Maxwell dataset

start date, duration, efforts actual, adjusted performance, estimated completion date, etc. [39].

The obtained results show that the Gravitational-based KRR with RBF Kernel method provides better performance compraing the Gravitational-based KRR with Wavelet Kernel method in this dataset. This is demonstrated is Table 6 and Fig 9.

5.6. **Comparison of the results.** We have applied the proposed GKRR to the Desharnais dataset in this section to evaluate its performance. The generated results by the GKRR and other feature selection-based regressors are given in Table 7. The reported results for GKRR are the average from different simulations. For each algorithm, we report the $MMRE$, $Pred(25)$, and

TABLE 6. Generated result in kitchenham dataset

Method	MMRE	Pred(25)	EF
Gravitational-based KRR with RBF Kernel	**0.2079**	**79.04**	**65.44**
Gravitational-based KRR with Wavelet Kernel	0.2146	78.86	64.93

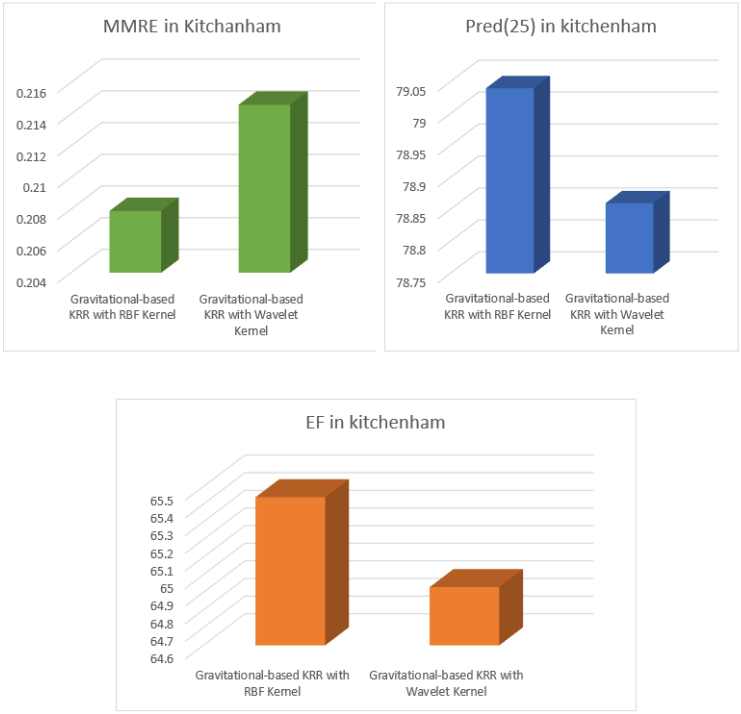

FIGURE 9. Comparison of the results obtained on kitchenham dataset

EF. This table indicates the superiority of GKRR comparing the other feature selection-based regressors in terms of $Pred(25)$, EF, and $MMRE$.

In Table 10, comparison of proposed method and other classical well-known regression techniques on the Albrecht dataset are given. This table shows that GKRR outperforms the competitive algorithms both in terms of $Pred(25)$ and EF.

Comparison of the other feature selection-based regressors and generated results on the Maxwell dataset are given in Table 8. This table indicates the superiority of GKRR comparing the other feature selection- based regressors in terms MMRE.

TABLE 7. Comparison of the different algorithms on the Desharnais dataset.

Method	$MMRE$	$Pred(25)$	EF
MLP [23]	0.7824	20.00	11.22
SVR [23]	0.7383	33.30	19.16
ANFIS [23]	1.0570	20.00	9.72
SVR with RBF Kernel [53]	0.4736	55.56	37.70
MLP [49]	0.4069	66.67	47.39
M5P regression tree [49]	0.6135	55.56	34.43
CBR [25]	0.3620	42.00	30.84
ANNcite [25]	0.3520	44.00	32.54
GP [16]	0.6230	51.60	31.79
HM-MLP-[Avg] [23]	0.6934	0.2670	15.77
HM-MLP-[WtAvg] [23]	0.8179	26.70	11.00
HM-MLP-[MLP] [23]	0.9079	20.00	13.99
HM-MLP-[SVR] [23]	0.7894	26.70	14.92
HM-MLP-[FIS-FCM] [23]	0.621	26.70	16.46
HM-MLP-[FIS-SC] [23]	0.6284	26.70	16.40
HM-MLP-[ANFIS-FCM] [23]	0.6205	26.70	16.48
HM-MLP-[ANFIS-SC] [23]	0.5901	26.70	25.16
HM-SVR-[Avg] [23]	0.5901	40.00	25.16
HM-SVR-[WtAvg] [23]	0.5903	40.00	25.15
HM-SVR-[MLP] [23]	0.8397	20.00	10.87
HM-SVR-[SVR] [23]	0.9303	26.70	13.83
HM-SVR-[FIS-FCM] [23]	0.9745	13.30	6.73
HM-SVR-[FIS-SC] [23]	0.8208	20.00	10.98
HM-SVR-[ANFIS-FCM] [23]	0.8197	20.00	10.99
HM-SVR-[ANFIS-SC] [23]	0.7657	26.70	15.12
HM-ANFIS-[Avg] [23]	1.1532	20.00	9.28
HM-ANFIS-[WtAvg] [23]	1.0588	20.00	9.71
HM-ANFIS-[MLP] [23]	1.0753	26.70	12.87
HM-ANFIS-[SVR] [23]	0.9080	26.70	13.99
HM-ANFIS-[FIS-FCM] [23]	1.0429	13.30	6.51
HM-ANFIS-[FIS-SC] [23]	0.9221	6.70	3.48
HM-ANFIS-[ANFIS-FCM] [23]	48.5060	13.30	0.26
HM-ANFIS-[ANFIS-SC] [23]	0.9154	13.30	6.94
HT-(MLP,SVR,ANFIS)-[Avg] [23]	0.6613	33.30	20.04
HT-(MLP,SVR,ANFIS)-[WtAvg] [23]	0.6686	26.70	16.00
HT-(MLP,SVR,ANFIS)-[MLP] [23]	0.7930	33.30	18.57
HT-(MLP,SVR,ANFIS)-[SVR] [23]	0.9249	26.70	13.87
HT-(MLP,SVR,ANFIS)-[FIS-SC] [23]	0.6360	20.00	12.22
HT-(MLP,SVR,ANFIS)-[FIS-FCM] [23]	0.8529	20.00	10.79
HT-(MLP,SVR,ANFIS)-[ANFIS-SC] [23]	0.6820	20.00	11.89
HT-(MLP,SVR,ANFIS)-[ANFIS-FCM] [23]	0.7614	40.00	22.71
Bagging (MLP) [49]	0.3991	66.67	47.65
Bagging (M5P) [49]	0.6054	55.56	34.61
ELMAN Neural Network [27]	0.5721	54.70	42.50
FCM [27]	0.4120	72.22	5140
SC [27]	0.7320	66.67	48.72
FSSS [7]	0.2870	72.22	54.90
GA-based SVR with RBF Kernel [48]	0.4051	61.11	38.33
GA-based SVR with Linear Kernel [48]	0.3685	–	6.20
GA-based MLP [48]	0.3154	–	7.00
GA-based M5P [48]	0.5945	–	0.00
SVR-PCA-GA [23]	–	–	–
MLP-PCA-GA [23]	–	–	–
ANFIS-PCA-GA [23]	–	13.33	–
PRED-MLP [37]	0.6347	40.00	–
PRED-MLP-FS [37]	0.2567	13.3330	–
FS+GA+MLP [10]	0.4776	52.1739	–
SVR – Boruta [66]	0.4570	34.7830	–
Gravitational-based KRR with RBF Kernel	0.2489	75.82	60.71
Gravitational-based KRR with Wavelet Kernel	**0.2145**	**78.24**	**64.42**

TABLE 8. Comparison of the different algorithms on the MAXWELL dataset.

Method	$MMRE$	$Pred(25)$	EF
MLP [23]	0.7840	25.00	33.00
SVR [23]	1.0131	33.30	33.00
ANFIS [23]	1.9049	16.7	9.00
CART [47]	–	32.00	–
HM-MLP-[Avg] [23]	0.5324	25.00	46.00
HM-MLP-[WtAvg] [23]	0.6603	16.7	25.00
HM-MLP-[MLP] [23]	1.0923	8.3	8.00
HM-MLP-[SVR] [23]	1.4149	16.7	12.00
HM-MLP-[FIS-FCM] [23]	0.8256	16.7	20.00
HM-MLP-[FIS-SC] [23]	0.6436	25.00	38.00
HM-MLP-[ANFIS-FCM] [23]	2.9691	33.33	1.00
HM-MLP-[ANFIS-SC] [23]	0.6053	33.33	54.00
HM-SVR-[Avg] [23]	0.7774	25.00	32.00
HM-SVR-[MLP] [23]	0.8526	33.33	39.00
HM-SVR-[WtAvg] [23]	1.0087	16.7	16.00
HM-SVR-[FIS-FCM] [23]	1.0643	16.7	16.00
HM-SVR-[SVR] [23]	1.3933	16.7	12.00
HM-SVR-[ANFIS-FCM] [23]	431.8376	0.00	0.00
HM-SVR-[FIS-SC] [23]	0.7892	25.00	31.00
HM-ANFIS-[Avg] [23]	0.6406	25.00	38.00
HM-SVR-[ANFIS-SC] [23]	0.6646	25.00	37.00
HM-ANFIS-[MLP] [23]	0.9225	8.30	9.00
HM-ANFIS-[WtAvg] [23]	0.6588	25.00	37.00
HM-ANFIS-[FIS-FCM] [23]	1.1937	8.30	7.00
HM-ANFIS-[SVR] [23]	1.3934	16.70	12.00
HM-ANFIS-[ANFIS-FCM] [23]	419.2451	8.30	0.00
HM-ANFIS-[FIS-SC] [23]	0.7525	25.00	33.00
HT-(MLP,SVR,ANFIS)-[Avg] [23]	0.6631	16.70	26.00
HM-ANFIS-[ANFIS-SC] [23]	0.5949	25.00	41.00
HT-(MLP,SVR,ANFIS)-[MLP] [23]	0.9619	8.30	9.00
HT-(MLP,SVR,ANFIS)-[WtAvg] [23]	0.6379	16.70	26.00
HT-(MLP,SVR,ANFIS)-[SVR] [23]	1.6708	16.70	10.00
HT-(MLP,SVR,ANFIS)-[FIS-SC] [23]	0.6234	25.00	39.00
HT-(MLP,SVR,ANFIS)-[FIS-FCM] [23]	1.1427	0.00	0.00
HT-(MLP,SVR,ANFIS)-[ANFIS-SC] [23]	0.5949	25.00	41.00
HT-(MLP,SVR,ANFIS)-[ANFIS-FCM] [23]	312.6672	8.30	0.00
SVR-PCA-GA [23]	–	–	**53.8**
MLP-PCA-GA [23]	–	–	19.3
ANFIS-PCA-GA [23]	–	13.33	5.9
3LEE [47]	–	**50.00**	–
ELMAN Neural Network [22]	1.3748	5.5556	2.34
FCM [22]	0.3706	38.88	28.37
SC [22]	0.5021	34.7830	18.49
NON-SMOOTHING [**?**]	0.477	–	–
2-ROUND SMOOTHING [**?**]	0.408	–	–
McFIS [52]	0.48	74.06	50.41
Gravitational-based KRR with RBF Kernel	**0.3351**	43.29	32.42
Gravitational-based KRR with Wavelet Kernel	0.3493	44.86	33.25

Comparison of the other feature selection-based regressors and generated results on the Kitchenham dataset are given in Table 9. This table indicates the superiority of GKRR comparing the other feature selection- based regressors in terms of Pred(25), EF , and MMRE.

TABLE 9. Comparison of the different algorithms on the Kitchenham dataset.

Method	$MMRE$	$Pred(25)$	EF
Polynomial Linear Regression [39]	6.2888	45.9459	6.3
Ridge Regression [39]	0.4182	40.54	28.59
Decision Tress [39]	37.25	540540	39.38
SVR [39]	–	16.216	2.7
MLP [39]	0.5324	16.216	2.7
ELMAN Neural Network [22]	55.95	22.22	14.25
FCM [22]	0.3706	15.5556	10.1
SC [22]	0.5021	8.8889	4.56
NON-SMOOTHING [**?**]	0.570	–	–
2-ROUND SMOOTHING [**?**]	0.574	–	–
McFIS [52]	0.36	78.4	57.65
CFS [45]	0.34	–	–
LFS [45]	0.28	–	–
LR [45]	0.34	–	–
mRMR [45]	0.3	–	–
Coff [54]	–	68.00	–
CBR [38]	–	21.8	–
ANN [38]	–	20.6	–
CART [38]	–	24.1	–
Gravitational-based KRR with RBF Kernel	**0.2079**	**0.7904**	**0.6544**
Gravitational-based KRR with Wavelet Kernel	0.2146	78.86	64.93

Based on the results obtained in Tables 7, 10, 8 and 9, it can be seen that the proposed method provides better performance than the methods introduced in reference [23] and the degree of accuracy is significantly different from these methods. The reason for this difference can be found in Table 1. As can be seen in this table, the methods presented in Reference [23] did not use any feature selection and parameter selection process. Therefore, in addition to the unrelated and noisy features in this model, it can be said that the best values of the parameters have not been selected. The same is true of the methods in references [16] and [66].

On the other hand, the method presented in reference [7] has a better performance than other previous methods. Because in this method, a feature selection approach has been used, which can be seen to improve it compared to the methods that did not use feature selection. However, this method was also less accurate than the proposed method. Because no parameter selection has been made in it. The method presented in Reference [65] also used the bee colony algorithm to select the parameter but did not record good accuracy due to the lack of selection of compelling features. However, the only methods

that have used feature selection and parameter selection simultaneously are the methods presented in references [48] and [10]. These methods also had much poorer performance than the proposed method. Among the reasons for this are the use of different primary learners and poor performance in selecting features and parameters.

6. Conclusions and future research

Software cost estimation is an important part of software project development. In this activity, required cost for software development and maintenance is predicted based on a wide range of features in the project environment. In most research work, the term software cost estimation is often considered to be equivalent to the software development effort estimation. If SCE is not precisely done at the beginning of project, the project may be doomed to fail in middle of its process. Feature selection task is an essential issue in creating prediction models. A smaller feature set makes the prediction decision more interpretable. This study presents a hybrid Gravitational Search Algorithm (GSA) method, with the ability to search for the optimal parameter values for Kernel Ridge Regression (KRR) to achieve a subset of useful features in the application of Software Development Effort Estimation (SDEE). This work is novel since, to the best of our knowledge, there is no empirical research performed on the hybrid GKRR prediction model to obtain model parameters and an optimal feature subset simultaneously. The performance of the KRR algorithm is assessed according to two benchmark datasets of software projects. The results indicate the superiority of the proposed algorithm over some recent methods in the recent literature for SDEE. Results of the GKRR were obtained with two RBF and Wavelet kernels. However, by the same approach, other kernel parameters can be optimized to extend and verify this technique in the future. Moreover, since the proposed GKRR is only applied to regression problems in SDEE, the efficiency of the proposed approach on the other regression problems and all classification problems will be examined in future research.

7. Aknowledgement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] R. de A. Araújo, A.L.I. Oliveira, S. Meira, *A class of hybrid multilayer perceptrons for software development effort estimation problems*, Artif. Intell. Rev vol., no. 90 (2017) 1–12.

[2] M. Abdel-Basset, W. Ding, D. El-Shahat, *A hybrid Harris Hawks optimization algorithm with simulated annealing for feature selection*, Expert Syst. Appl vol., no. 54 (2021) 593–637.

TABLE 10. Comparison of the competitive algorithms on the Albrecht dataset.

Method	$MMRE$	$Pred(25)$	EF
MLP [23]	0.6460	40.00	24.30
SVR [23]	0.8140	20.00	11.03
ANFIS [23]	0.7861	20.00	11.20
GP [16]	0.5480	64.00	41.34
HM-MLP-[Avg] [23]	**0.3299**	20.00	15.04
HM-MLP-[WtAvg] [23]	0.4413	40.00	27.75
HM-MLP-[MLP] [23]	0.5423	40.00	25.94
HM-MLP-[SVR] [23]	0.7438	40.00	22.94
HM-MLP-[FIS-FCM]	0.6143	20.00	12.39
HM-MLP-[ANFIS-FCM] [23]	0.7252	40.00	23.19
HM-SVR-[FIS-SC] [23]	0.4387	40.00	27.80
HM-SVR-[Avg] [23]	0.6765	20.00	11.93
HM-MLP-[ANFIS-SC] [23]	0.6304	20.00	12.27
HM-SVR-[MLP] [23]	0.6930	20.00	11.81
HM-SVR-[WtAvg] [23]	0.6802	20.00	11.90
HM-SVR-[FIS-FCM] [23] [23]	0.7752	20.00	11.27
HM-SVR-[SVR] [23]	0.7425	40.00	22.96
HM-SVR-[ANFIS-FCM] [23]	0.4863	60.00	40.37
HM-SVR-[FIS-SC] [23]	0.9458	0.00	0.00
HM-ANFIS-[Avg]	0.7588	0.00	0.00
HM-SVR-[ANFIS-SC] [23]	0.3820	60.00	43.42
HM-ANFIS-[MLP] [23]	0.6816	40.00	23.79
HM-ANFIS-[WtAvg] [23]	0.6943	20.00	11.80
HM-ANFIS-[FIS-FCM] [23]	0.6724	20.00	11.96
HM-ANFIS-[SVR] [23]	0.8083	40.00	22.12
HM-ANFIS-[ANFIS-FCM] [23]	0.630	20.00	11.95
HM-ANFIS-[FIS-SC] [23]	1.0883	0.00	0.00
HT-(MLP,SVR,ANFIS)-[Avg] [23]	0.8479	0.00	0.00
HM-ANFIS-[ANFIS-SC] [23]	1.3601	0.00	0.00
HT-(MLP,SVR,ANFIS)-[MLP] [23]	0.9203	20.00	11.54
HT-(MLP,SVR,ANFIS)-[WtAvg] [23]	0.7330	20.00	10.66
HT-(MLP,SVR,ANFIS)-[SVR] [23]	0.8770	20.00	10.42
HT-(MLP,SVR,ANFIS)-[FIS-SC] [23]	0.8677	0.00	0.00
HT-(MLP,SVR,ANFIS)-[FIS-FCM] [23]	1.0682	0.00	0.00
HT-(MLP,SVR,ANFIS)-[ANFIS-SC] [23]	0.5882	20.00	12.59
HT-(MLP,SVR,ANFIS)-[ANFIS-FCM] [23]	0.6059	20.00	12.45
GA-based SVR with RBF Kernel [48]	0.4465	70.42	**48.68**
GA-based SVR with Linear Kernel [48]	0.6628	56.25	33.88
GA-based M5 [48]	0.4700	45.83	31.7
GA-based MLP [48]	0.6863	61.67	22.71
SVR-PCA-GA [23]	–	–	0.3649
MLP-PCA-GA [23]	–	–	0.00
ANFIS-PCA-GA [23]	–	–	24.50
FS+GA+MLP [10]	0.4038	42.8571	0.00
SVR – Boruta [65]	0.5660	28.57	–
GP [66]	0.5549	58.333	–
Gravitational-based KRR with RBF Kernel	0.7322	62.50	36.08
Gravitational-based KRR with Wavelet Kernel	0.6723	**72.83**	43.61

[3] R. Abu Khurmaa, I. Aljarah, A. Sharieh, *An intelligent feature selection approach based on moth flame optimization for medical diagnosis*, Neural Comput. Appl vol., no. 33 (2021) 7165–7204.

[4] R. Ahila, V. Sadasivam, K. Manimala, *An integrated PSO for parameter determination and feature selection of ELM and its application in classification of power system disturbances*, Appl. Soft Comput vol., no. 32 (2015) 23–37.

[5] B. Ahuja, V.P. Vishwakarma, *Deterministic Multi-kernel based extreme learning machine for pattern classification*, Expert Syst. Appl vol., no. 183 (2021) 115308.

[6] M. Al Asheeri, M. Hammad, *Improving software cost estimation process using feature selection technique*, 3rd Smart Cities Symp, (2021), 89–95.

[7] M. Azzeh, D. Neagu, P. Cowling, *Improving analogy software effort estimation using fuzzy feature subset selection algorithm*, 4th Int. Work. Predict. Model. Softw. Eng, (New York, 2008), 71–78.

[8] V.K. Bardsiri, D.N.A. Jawawi, A.K. Bardsiri, E. Khatibi, *LMES: A localized multi-estimator model to estimate software development effort*, Eng. Appl. Artif. Intell vol., no. 26 (2013) 2624–2640.

[9] B.W. Boehm, *Software Engineering Economics*, IEEE Trans. Softw. Eng. Appl vol., no. 10 (1984) 4–21.

[10] P.L. Braga, A.L.I. Oliveira, S.R.L. Meira, *A GA-based feature selection and parameters optimization for support vector regression applied to software effort estimation*, Proc. 2008 ACM Symp. Appl. Comput, (2008) p. 1788.

[11] L.C. Briand, I. Wieczorek, *Resource Estimation in Software Engineering*, Encycl. Softw. Eng vol., no. 2 (2021) 1160–1196.

[12] B. Charbuty, A. Abdulazeez, *Classification Based on Decision Tree Algorithm for Machine Learning*, J. Appl. Sci. Technol. Trends vol., no. 2 (2021) 20–28.

[13] E. D, B. B, F. T, D. J, U. J, *Parametric estimating handbook*, The International Society of Parametric Analysis (ISPA), 2009.

[14] P. Decker, R. Durand, C. O. Mayfield, C. McCormack, D. Skinner, G. Perdue, *Predicting implementation failure in organization change*, Cult. Commun. Confl vol., no. 16 (2012) 29.

[15] J.-M. Desharnais, *Statistical analysis on the productivity of data processing with development projects using the function point technique*, Université du Québec à Montréal, 1988.

[16] J.. Dolado, *On the problem of the software cost function*, Inf. Softw. Technol vol., no. 43 (2001) 61–72.

[17] H. Dong, L. Yang, *Kernel-based regression via a novel robust loss function and iteratively reweighted least squares*, Knowl. Inf. Syst vol., no. 43 (2001) 61–72.

[18] M.B. Dowlatshahi, V. Derhami, H. Nezamabadi-Pour, *Fuzzy particle swarm optimization with nearest-better neighborhood for multimodal optimization*, Iran. J. Fuzzy Syst. vol., no.6317 (2021) 1149–1172.

[19] M.B. Dowlatshahi, M. Kuchaki Rafsanjani, B.B. Gupta, *An energy aware grouping memetic algorithm to schedule the sensing activity in WSNs-based IoT for smart cities*, Appl. Soft Comput vol., no. 108 (2021) 107473.

[20] M.B. Dowlatshahi, H. Nezamabadi-Pour, *GGSA: A Grouping Gravitational Search Algorithm for data clustering*, Eng. Appl. Artif. Intell vol., no. 36 (2014) 114–121.

[21] M.B. Dowlatshahi, H. Nezamabadi-Pour, M. Mashinchi, *A discrete gravitational search algorithm for solving combinatorial optimization problems*, Inf. Sci vol., no. 258 (2014) 94–107.

[22] P. Edinson, L. Muthuraj, *Performance Analysis of FCM Based ANFIS and ELMAN Neural Network in Software Effort Estimation*, Int. Arab J. Inf. Technol vol., no. 15 (2018).

[23] M.O. Elish, T. Helmy, M.I. Hussain, *Empirical Study of Homogeneous and Heterogeneous Ensemble Models for Software Development Effort Estimation*, Math. Probl. Eng vol., no. 2013 (2013) 1–21.
[24] F.-L. Fan, J. Xiong, M. Li, G. Wang, *On Interpretability of Artificial Neural Networks: A Survey*, IEEE Trans. Radiat. Plasma Med. Sci (2021) 741–760.
[25] G.R. Finnie, G.E. Wittig, J.-M. Desharnais, *A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models*, J. Syst. Softw vol., no. 39 (1997) 281–289.
[26] Galorath, D. D, M.W. Evans, *Software sizing, estimation, and risk management: when performance is measured performance improves*, Auerbach Publications, 2006.
[27] M.R. Garey, D.S. Johnson, *COMPUTERS AND INTRACTABILITY: A Guide to the Theory of NP-Completeness*, San Francisco: freeman, 1979.
[28] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, Rui Zhang *Variable Neighborhood Search*, Handbook of metaheuristics. Springer, Cham (2019) 57–97.
[29] P. Hansen, N. Mladenović, J. Brimberg, J.A.M. Pérez, *A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models*, J. Syst. Softw vol., no. 39 (1997) 281–289.
[30] A. Hashemi, M. Bagher Dowlatshahi, H. Nezamabadi-pour, *VMFS: A VIKOR-based multi-target feature selection*, Expert Syst. Appl (2021) 115224.
[31] A. Hashemi, M. Bagher Dowlatshahi, H. Nezamabadi-pour, *MFS-MCDM: Multi-label feature selection using multi-criteria decision making*, Knowledge-Based Syst (2020) 106365.
[32] A. Hashemi, M.B. Dowlatshahi, H. Nezamabadi-pour, *Gravitational Search Algorithm, in: Handb. AI-Based Metaheuristics*, CRC Press, 2021.
[33] A. Hashemi, M. Bagher Dowlatshahi, H. Nezamabadi-pour, *Ensemble of feature selection algorithms: a multi-criteria decision-making approach*, Int. J. Mach. Learn. Cybern vol., no. 13 (2022) 49–69.
[34] A. Hashemi, M. Bagher Dowlatshahi, H. Nezamabadi-pour *A bipartite matching-based feature selection for multi-label learning*, Int. J. Mach. Learn. Cybern vol., no. 12 (2020) 459–475.
[35] P. He, J.-K. Hao, *Iterated two-phase local search for the colored traveling salesmen problem*, Eng. Appl. Artif. Intell vol., no. 97 (2021) 104018.
[36] D.E. Holland, R.J. Olesen, J.E. Bevins, *Multi-objective genetic algorithm optimization of a directionally sensitive radiation detection system using a surrogate transport model*, Eng. Appl. Artif. Intell. Cybern vol., no. 104 (2021) 104357.
[37] J.H. Holland, *Outline for a Logical Theory of Adaptive Systems*, J. ACM vol., no. 9 (1962) 297–314.
[38] J. Huang, Y.-F. Li, M. Xie, *An empirical analysis of data preprocessing for machine learning-based software cost estimation*, Inf. Softw. Technol vol., no. 67 (2015) 108–127.
[39] R. Israr Ur, A. Zulfiqar, J. Zahoor, *An Empirical Analysis on Software Development Efforts Estimation in Machine Learning Perspective*, ADCAIJ Adv. Distrib. Comput. Artif. Intell. J vol., no. 10 (2021) 227–240.
[40] G. Bin Huang, C.K. Slew, *Extreme learning machine: RBF network case*, 8th Int. Conf. Control. Autom. Robot. Vis., IEEE, (Kunming, 2004), 1029–1036.
[41] A. Arabipour, M. Amini, *A weighted linear regression model for impercise response*, J. Mahani Math. Res vol., no. 3(2014), 1–17.
[42] K. Korenaga, A. Monden, Z. Yucel, *Data Smoothing for Software Effort Estimation*, 20th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput, (Toyama, 2019), 501–506.
[43] C. Li, X. An, R. Li, *A chaos embedded GSA-SVM hybrid system for classification*, Neural Comput. Appl vol., no. 26 (2015) 713–721.

[44] B. Liang, Y. Zhao, Y. Li, *A hybrid particle swarm optimization with crisscross learning strategy*, Eng. Appl. Artif. Intell vol., no. 105 (2021) 104418.
[45] Q. Liu, J. Xiao, H. Zhu, *Feature selection for software effort estimation with localized neighborhood mutual information*, Cluster Comput vol., no. 22 (2019) 6953–6961.
[46] P. MacDonell, Stephen; Whigham, *Data Quality in Empirical Software Engineering: An Investigation of Time-Aware Models in Software Effort Estimation*, University of Otago (2016) 1155–1166.
[47] A. Moradbeiky, V. Khatibi, M. Jafari Shahbazzadeh, *3LEE: A 3-Layer Effort Estimator for Software Projects*, Int. J. Ind. Electron. Control Optim vol., no. 5 (2022) 31–42.
[48] A.L.I. Oliveira, P.L. Braga, R.M.F. Lima, M.L. Cornélio, *GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation*, Inf. Softw. Technol vol., no. 52 (2010) 1155–1166.
[49] K. ONO, M. TSUNODA, A. MONDEN, K. MATSUMOTO, *Influence of Outliers on Estimation Accuracy of Software Development Effort*, IEICE Trans. Inf. Syst vol., no. 104 (2021) 91–105.
[50] M. Paniri, M.B. Dowlatshahi, H. Nezamabadi-pour, *Ant-TD: Ant colony optimization plus temporal difference reinforcement learning for multi-label feature selection*, Swarm Evol. Comput vol., no. 64 (2021) 713–721.
[51] D.A. Pisner, D.M. Schnyer, *Support vector machine*, Mach. Learn., Elsevier, 2020.
[52] E. Praynlin, *Using meta-cognitive sequential learning Neuro-fuzzy inference system to estimate software development effort*, J. Ambient Intell. Humaniz. Comput vol., no. 12 (2021) 8763–8776.
[53] L.H. Putnam, *A General Empirical Solution to the Macro Software Sizing and Estimating Problem*, IEEE Trans. Softw. Eng vol., no. 4 (1978) 345–361.
[54] F. Qi, X.-Y. Jing, X. Zhu, X. Xie, B. Xu, S. Ying, *Software effort estimation based on open source projects: Case study of Github*, Inf. Softw. Technol vol., no. 92 (2017) 145–157.
[55] C.R. Rao, *GENERALIZED INVERSE OF A MATRIX AND ITS APPLICATIONS*, Theory Stat., University of California Press, 1972.
[56] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, *GSA: A Gravitational Search Algorithm*, Inf. Sci vol., no. 179 (2009) 2232–2248.
[57] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, *Filter modeling using gravitational search algorithm*, Eng. Appl. Artif. Intell vol., no. 24 (2011) 117–122.
[58] M. Relich, P. Pawlewski, *A case-based reasoning approach to cost estimation of new product development*, Neurocomputing vol., no. 272 (2018) 40–45.
[59] S.H. Samareh Moosavi, V. Khatibi Bardsiri, *Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation*, Eng. Appl. Artif. Intell vol., no. 60 (2017) 1–15.
[60] J.S. Sartakhti, M.H. Zangooei, K. Mozafari, *Hepatitis disease diagnosis using a novel hybrid method based on support vector machine and simulated annealing (SVM-SA)*, Comput. Methods Programs Biomed vol., no. 108 (2012) 570–579.
[61] S. Sundaram, P. Kellnhofer, Y. Li, J.-Y. Zhu, A. Torralba, W. Matusik, *Learning the signatures of the human grasp using a scalable tactile glove*, Nature vol., no. 569 (2019) 698–702.
[62] P. Suresh Kumar, H.S. Behera, A.K. K, J. Nayak, B. Naik, *Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades*, Comput. Sci. Rev vol., no. 38 (2020) 100288.
[63] Z. Tao, L. Huiling, W. Wenwen, Y. Xia, *GA-SVM based feature selection and parameter optimization in hospitalization expense modeling*, Appl. Soft Comput vol., no. 75 (2019) 323–332.
[64] B. Venkatesh, J. Anuradha, *A review of Feature Selection and its methods*, Cybern. Inf. Technol vol., no. 19 (2019) 3–26.

[65] Z.H. Wani, S.M.K. Quadri, *Artificial Bee Colony-Trained Functional Link Artificial Neural Network Model for Software Cost Estimation*, Proceedings of Fifth International Conference on Soft Computing for Problem Solving. Springer, Singapore, (2016) 729–741.
[66] J. Wen, S. Li, Z. Lin, Y. Hu, C. Huang, *A Systematic literature review of machine learning based software development effort estimation models*, Inf. Softw. Technol vol., no. 54 (2012) 41–59.
[67] A. ZAKRANI, M. HAIN, A. IDRI, *A review of Feature Selection and its methods*, IAES Int. J. Artif. Intell vol., no. 8 (2019) 3–26.
[68] N. Zeng, H. Qiu, Z. Wang, W. Liu, H. Zhang, Y. Li, *A review of Feature Selection and its methods*, Neurocomputing vol., no. 320 (2018) 195–202.
[69] N. Zeng, H. Qiu, Z. Wang, W. Liu, H. Zhang, Y. Li, *Feature selection with multi-view data: A survey*, Inf. Fusion vol., no. 50 (2019) 158–167.
[70] Y. Zhou, J.-K. Hao, *Tabu search with graph reduction for finding maximum balanced bicliques in bipartite graphs*, Eng. Appl. Artif. Intell vol., no. 77 (2019) 86–97.

Mohammad Bagher Dowlatshahi
Orcid number: 0000-0002-4862-0626
Department of Computer Engineering
Lorestan University
Khorramabad, Iran
Email address: dowlatshahi.mb@lu.ac.ir

Mohammad Ali Zare-Chahooki
Orcid number: 0000-0003-3900-6656
Department of Computer Engineering
Yazd University
Yazd, Iran
Email address: chahooki@yazd.ac.ir

Saba Beiranvand
Orcid number: 0000-0002-3468-2981
Department of Computer Engineering
Technical and Vocational University (TVU)
Tehran , Iran
Email address: bir.saba@yahoo.com

Amin Hashemi
Orcid number: 0000-0002-3768-0615
Department of Computer Engineering
Lorestan University
Khorramabad, Iran
Email address: hashemi.am@fe.lu.ac.ir