Cite this: DOI: 00.0000/xxxxxxxxxx

A machine learning methodology for reliability evaluation of complex chemical production systems

Fanrui Zhao,[a] Jinkui Wu,[a] Yuanpei Zhao,[b] Xu Ji,[a] Li Zhou, *[a] and Zhongping Sun [a]

Received Date
Accepted Date

DOI: 00.0000/xxxxxxxxxx

Supplementary Material

1 Grey Relational Analysis (GRA)

The grey relational analysis method is part of the gray system theory, which was first proposed by Professor Deng Julong of Huazhong University of Science and Technology in 1982[1]. The grey system theory studies the modeling of poor information systems with little data. Grey relational analysis method is used to describe the consistency problem of the change trend of two factors, namely the relevance between each sub-sequence and the parent sequence, and to compare the correlation. It has been proved to be effective in solving problems with complicated interrelationships between multiple factors and variables[2]. Classical GRA is based on time series data and/or cross-sectional data.

The analysis steps of the grey relational analysis method are as follows[3]: First, the grey relation is generated, that is, all sequences reflecting the behavioral characteristics of the system are converted into comparison sequences; then a reference sequence is defined in these sequences, and calculate the grey relational coefficient between all the comparison sequences and the reference sequence; Finally, based on these correlation coefficients, calculate the grey correlation grade between each comparison sequence and the reference sequence, the comparison sequence with the highest correlation grade is the best choice.

(1)Generate grey correlation

When the dimensions of the attributes are not uniform or the magnitude differs too much, the influence of some attributes may be ignored. Therefore, it is necessary to use a method similar to normalization to convert each attribute into a comparable sequence. This process is called grey correlation generation. This paper adopts the averaging method to deal with the problem of data dimension, so as to normalize the data.

(2)Define the reference sequence

After grey correlation generation, the normalization of data dimension is completed. Next, determine the reference sequence that reflects the characteristics of the system's behavior and the comparison sequence that affects the system's behavior. Set reference sequence is $(X_0(k), k = 1,2,\cdots,n)$, compare sequence is $(X_i = X_i(k)|k = 1,2,\cdots,n, i = 1,2,\cdots,m)$.

(3)Calculate the grey correlation coefficient

The grey correlation coefficient reflects the correlation between the elements in the comparison sequence and the $X_i(k)$ elements in the reference sequence $X_0(k)$. The calculation formula is as follows:

$$\xi_i(k) = \frac{\min\limits_i \min\limits_k |X_0(k) - X_i(k)| + \rho \max\limits_i \max\limits_k |X_0(k) - X_i(k)|}{|X_0(k) - X_i(k)| + \rho \max\limits_i \max\limits_k |X_0(k) - X_i(k)|} \quad (1)$$

where ρ is the resolution coefficient, its size can control the influence of the maximum difference on data conversion, which is generally 0.5.

(4)Calculate grey correlation grade

Finally, the average value of the correlation coefficient of each element is used as the grey correlation grade between the comparison sequence and the reference sequence. The formula is as follows:

$$r_i = \frac{1}{n} \sum_{k=1}^{n} \xi_i(k) \quad (2)$$

r_i is the grey correlation grade, the closer the value is to 1, the higher the degree of correlation.

2 Particle Swam Optimization (PSO)

Particle Swam Optimization (PSO) was first proposed by Kennedy and Eberhart in 1995[4]. Inspired by the bird's predatory behavior, PSO realized the process of finding the best solution in a complex space through cooperation and competition among individuals[5]. The algorithm involves fewer concepts and is easier to implement. A large number of studies have shown that the particle swarm optimization algorithm can solve nonlinear optimization problems

[a] Department of Chemical Engineering, College of Chemical Engineering, Sichuan University, Chengdu 610065, China; Tel: +86 15228867167; E-mail: chezli@scu.edu.cn
[b] State Key Laboratory of Power Transmission Equipment & System Security and New Technology, School of Electrical Engineering, Chongqing University, Chongqing, 400044, China

and combinatorial optimization problems well[6–8].

The core idea of the PSO algorithm is to simulate the predation behavior of birds. The individuals in the population are particles without volume and mass. The state of each particle represents a certain possible solution, and the optimal position of the population is the global optimal solution. With m particles, its population can be expressed as $Y = Y_1, Y_2, \cdots, Y_i, \cdots, Y_m$, and its D-dimensional attribute can be expressed as $Y_i = y_{i1}, y_{i2}, \cdots, y_{iD}$. The optimal position of the individual itself is called the individual optimal value P_i , and the optimal position of the population is called the global optimal value P_g . The state of the individual is determined by the speed $V_i = v_{i1}, v_{i2}, \cdots, v_{iD}$ and the optimal position $P_i = p_{i1}, p_{i2}, \cdots, p_{iD}$, then the state of the particle can be updated by Eq.3 and Eq.4. The fitness value of Y_i is determined by calculating the fitness function (y_{ik}) of each particle, and the individual extreme values P_i^k and global extreme values P_g^k are updated by comparison with the previous one, thereby generating a new generation of population. When the termination condition is satisfied (usually the maximum number of iterations or the precision that needs to be reached), the iteration stops and a global optimal solution is produced.

$$V_{i,d}^{(k+1)} = \omega V_{i,d}^k + c_1 r_1 (P_{i,d}^{ind} - P_{i,d}^k) + c_2 r_2 (P_d^{glob} - P_{i,d}^k) \tag{3}$$

$$P_{i,d}^{(k+1)} = P_{i,d}^k + V_{i,d}^{(k+1)} \tag{4}$$

where k denotes the iteration number, d represents the search direction, ω is called inertial weight, r_1 and r_2 are random numbers with uniform distribution in the range of 0 to 1, and c_1 and c_2 are the cognition and social parameters respectively.

3 Support Vector Machine (SVM)

Support Vector Machine (SVM) was officially proposed by Cortes and Vapnik in 1995[9] . Its basic model is a linear classifier with the largest interval defined in the feature space, that is, it hopes to obtain a hyperplane through training to correctly separate positive and negative samples, and to ensure that the interval between the sample and the classification hyperplane obtained by training is the largest. Since support vector machines have multiple kernel functions, they can convert features in low dimensions into features with high dimensions for calculation, which is essentially equivalent to implicitly learning linear support vector machines in high-dimensional feature spaces, so therefore, it is essentially a nonlinear classifier. This model compromises the model load and accurate classification ability in exchange for stronger generalization ability, and can better solve the problems of small samples, nonlinearity, high dimensionality, and local minimum[10,11]. According to different problems, it can be divided into support vector machine for classifier (SVC) and support vector machine for regression (SVR).

3.1 Support vector machine for classifier (SVC)

(1)Linear classification problem

Consider the second-class classification problem. Suppose that given a training set sample $T = (x_1, y_1), (x_2, y_2), \cdots, (x_l, y_l), x_i \in R^n$, where each sample x_i has n-dimensional feature vectors, also known as instances, $y_i \in +1, -1$, y_i represents the label value of each sample, because it is a binary classification problem, there are only two values: When $y = +1$, it is called a positive case; when $y = -1$, it is a negative case. The goal of learning is to find a separate hyperplane in the feature space, which can divide the examples into different classes and maximize the interval of the hyperplanes.

Let the classification hyperplane be:

$$\omega \cdot x + b = 0 \tag{5}$$

The decision function is:

$$f(x) = sign(\omega \cdot x + b) \tag{6}$$

Where ω represents the normal vector of the hyperplane and b represents the intercept.

It can be seen from Eq.5 that that when ω and b change in the same proportion, the hyperplane represented does not change. It is possible to set the distance of the positive and negative samples from the hyperplane to at least 1 unit length, so we can get:

$$\begin{aligned} \omega \cdot x + b &\geq 1 \\ \omega \cdot x + b &\leq -1, i = 1, 2, 3, \cdots, l \end{aligned} \tag{7}$$

When $y = 1$, the sample is a positive sample, on the side of the hyperplane, and when $y = -1$, it is expressed as a negative sample, on the other side of the classification hyperplane. By maximizing the interval to train the SVM model, the optimal separation hyperplane can be finally obtained.

The final optimization goal is to maximize the value of $\frac{2}{\|\omega\|}$, considering the model generalization ability and model misclassification, and at the same time to minimize the empirical risk, so on the basis of $\frac{2}{\|\omega\|}$, a penalty term composed of relaxation variable ξ_i and penalty factor C is added. When the value of C is relatively large, the penalty for model misclassification increases. The relaxation variable ξ_i is also added to Eq.7, and the final equivalent constraint is $y_i(\omega \cdot x_i + b) \geq 1 - \xi_i$. Therefore, the solution of the SVM model can be converted into a convex quadratic programming problem with linear constraints. The optimization goal in Eq.8 requires not only the maximization of the interval but also the model misclassification as small as possible.

$$\begin{aligned} \min_{\omega, b, \xi} \quad & \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{l} \xi_i \\ s.t. \quad & y_i(\omega \cdot x_i + b) \geq 1 - \xi_i, i = 1, 2, 3, \cdots, l \\ & \xi_i \geq 0, i = 1, 2, 3, \cdots, l \end{aligned} \tag{8}$$

Since the dual problem is easier to solve and the kernel function can be directly introduced, the optimization problem is solved by the method of solving the dual problem. Introducing the Lagrangian multiplier $\alpha_i \geq 0, \beta_i \geq 0$, the Lagrange function is obtained:

$$L(\omega,b,\xi,\alpha,\beta)=\frac{1}{2}\|\omega\|^2+C\sum_{i=1}^{l}\xi_i-\sum_{i=1}^{l}\alpha_i[y_i(\omega\cdot x_i+b)-1+\xi_i]-\sum_{i=1}^{l}\beta_i\xi_i \tag{9}$$

According to Eq.9, find the partial derivative of ω,b,ξ to get the minimum value of L:

$$\frac{\partial L}{\partial \omega}=\omega-\sum_{i=1}^{l}\alpha_i y_i x_i=0 \tag{10}$$

$$\frac{\partial L}{\partial b}=-\sum_{i=1}^{l}\alpha_i y_i=0 \tag{11}$$

$$\frac{\partial L}{\partial \xi_i}=C-\alpha_i-\beta_i=0 \tag{12}$$

Substituting the results of Eq.11 and Eq.12 into Eq.9 and simplifying:

$$\begin{aligned}L&=\frac{1}{2}(\sum_{i=1}^{l}\alpha_i y_i x_i)^2+C\sum_{i=1}^{l}\xi_i-(\sum_{i=1}^{l}\alpha_i y_i x_i)^2+\sum_{i=1}^{l}\alpha_i-\sum_{i=1}^{l}(alpha_i+\beta_i)\xi_i\\&=-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i y_i x_i\alpha_i y_i x_i+\sum_{i=1}^{l}\alpha_i\end{aligned} \tag{13}$$

The resulting dual optimization problem is shown in Eq.14:

$$\begin{aligned}\min_{\alpha,\beta}\quad&\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i y_i x_i\alpha_i y_i x_i-\sum_{i=1}^{l}\alpha_i\\s.t.\quad&0\le\alpha_i\le C\\&\sum_{i=1}^{l}\alpha_i y_i=0\end{aligned} \tag{14}$$

Let $\alpha^*=]\alpha_1,\alpha_2,\alpha_3,\cdots,\alpha_l]$ be the optimal solution of the above convex optimization problem, then Eq.15 can be obtained according to the KKT condition, and it can be obtained according to the constraints of the dual problem: when $0\le\alpha_i\le C$, $y_i(\omega^*\cdot x_i+b)-1+\xi_i^*=0$, the corresponding sample point is called support vector.

$$\begin{aligned}y_i(\omega^*\cdot x_i+b)-1+\xi_i^*=0\\\beta_i^*\xi_i^*=0\end{aligned} \tag{15}$$

According to the above analysis, the value of ω, b and the classification decision function can be finally obtained:

$$\begin{aligned}\omega&=\sum_{i=1}^{l}\alpha_i y_i x_i\\b&=y_j-\sum_{i=1}^{l}\alpha_i y_i(x_i\cdot x_j)\qquad 0\le\alpha_i\le C\end{aligned} \tag{16}$$

$$f(x)=sign(\sum_{i=1}^{l}\alpha_i y_i(x_i\cdot x_j)+b) \tag{17}$$

(2)Nonlinear classification problem

When the samples are linearly inseparable, the support vector classifier solves this problem by converting the sample values into a linear feature space of high or even infinite dimensions through a nonlinear function $\phi(\cdot)$, then classify the samples in a high-dimensional space[12]. The decision function at this time is shown in Eq.18 , where $K(x_i,x_j)=\phi(x_i)\cdot\phi(x_j)$ is called the kernel function. There are currently four types of kernel functions commonly used:

$$\begin{aligned}f(x)&=sign(\sum_{i=1}^{l}\alpha_i y_i(\phi(x_i)\cdot\phi(x_j))+b)\\&=sign(\sum_{i=1}^{l}\alpha_i y_i K(x_i,x_j))+b)\end{aligned} \tag{18}$$

(a) Linear kernel function: $K(x_i,x_j)=x_i\cdot x_j$

(b) Polynomial kernel function: $K(x_i,x_j)=[(x_i\cdot x_j)+1]^d, d=1,2,3,\cdots,n$

(c) Radial basis kernel function: $K(x_i,x_j)=\exp(-\frac{|x_i-x|^2}{2\sigma^2})=exp(-\gamma|x_i-x_j|^2)$

3.2 Support vector machine for regression (SVR)

(1) Linear regression problem

Support vector machine were originally suitable for classification problems, but as research found that they were also suitable for regression problems, support vector regression machines came into being. SVR is a regression method based on penalty learning[13]. The difference between SVM and SVR is that the output value of the classification problem is $y_i\in+1,-1$, while the output value of the regression problem is any continuous value. The overall goal of SVR is to get the regression relationship between the input x and the result y. $f(x)$ can be described as an approximate linear regression problem.

$$y=f(x)=\sum_{i=1}^{l}\omega_i x_i+b \tag{19}$$

Among them, the regression coefficients ω and b are obtained by the optimization method.

The essence of the SVR optimization problem is the problem of minimizing the width $\frac{1}{2}\|\omega\|^2$ of the plane. In the presence of errors, the slack variables ξ^- and ξ^+ are introduced to denote the cases of not exceeding and exceeding the penalty interval ε respectively, and add to the parameter C constitute the final risk function R. The goal of the SVR problem is to convert the problem of minimizing the risk function

$$\begin{aligned}Min\cdot R(\omega,\xi^+,\xi^-)&=\frac{1}{2}\|\omega\|^2+C\sum_{i=1}^{n}(\xi^++\xi^-)\\s.t.\quad&f(x_i)-y_i\le\varepsilon+\xi^-,i=1,\cdots,n\\&y_i-f(x_i)\ge\varepsilon+\xi^+,i=1,\cdots,n\\&\xi^+,\xi^-\ge0,i=1,\cdots,n\end{aligned} \tag{20}$$

By introducing Lagrange multiplier α_i^- and α_i^+ , the above for-

mula is transformed into the following form:

$$Max \cdot L_q = \sum_{i=1}^{l}(\alpha_i^+ - \alpha_i^-)y_i - \varepsilon\sum_{i=1}^{l}(\alpha_i^+ + \alpha_i^-) - \frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_i^-)x_i y_j$$

$$s.t. \qquad 0 \le \alpha_i^+ \le C, i = 1, \cdots, l$$

$$0 \le \alpha_i^- \le C, i = 1, \cdots, l \tag{21}$$

Therefore, the initial regression goal can be expressed as:

$$y = f(x, \alpha^+, \alpha^-) = \sum_{i=1}^{l}(\alpha_i^+ - \alpha_i^-)x_i \cdot x + b \tag{22}$$

(2) Nonlinear regression problem

In the regression problem, solving the nonlinear problem is similar to solving the nonlinear classification problem. The kernel function is used to map the low-dimensional data samples to the high-dimensional space, so that the nonlinear problem is converted into a linear problem in the high-dimensional space, thereby achieving the nonlinear problem regression prediction. The choice of kernel function determines the accuracy and universality of the SVR regression model.

4 Random Forest (RF)

Random forest is an integrated learning method. The integrated learning method is a method that integrates the individual learner into a strong learner through a certain strategy to complete the learning task. Random forest is a combination algorithm of Bagging integration method based on decision tree. The following will introduce the integrated learning method, decision tree , Bagging and random forest.

4.1 Integrated learning

Integrated learning accomplishes the learning task by constructing multiple learners. The structure of integrated learning is generally: first determine a group of individual learners, and then combine them into a combined module through a certain strategy, and finally the combined module completes the learning task and outputs it. Individual learners are generally existing learning algorithms. If the individual learners are all the same, such as "neural network integration" is all integrated by a neural network, it is called homogeneous integration, and the individual learner in homogeneous integration is called "basic learning". If the individual learner is composed of different algorithms, such as the individual learner contains both a decision tree and a neural network, it is called heterogeneous integration, and the individual learner in heterogeneous integration is generally called "organization learner" . Integrated learning achieves better generalization performance than a single learner by integrating multiple individual learners, especially compared to the "weak learner", therefore, the basic learner is often called weak learner, and the combined learner is also often called strong learner.

According to the different combination strategies of individual learners, integrated learning is divided into two integration methods. The first one is Boosting. In this method, the individual learners are related to each other and must be integrated in series, that is, the latter learner must be based on the previous learning results; the second one is Bagging, there is no correlation between each learner in this method, they can learn in parallel at the same time. Random forest is a learning method based on Bagging.

4.2 Decision tree

Decision tree is a basic classification regression model. This idea mainly comes from the ID3 algorithm in 1986 and the C4.5 algorithm proposed in 1993 proposed by Quinlan, and the CART algorithm proposed by Breinman et al. in 1984[14–16]. The decision tree consists of nodes and directed edges. Internal nodes represent a certain feature, and leaf nodes represent a certain category. When using the decision tree for classification, starting from the top root node, the attributes of the sample are discriminated one by one, and classified into an internal node representing the corresponding attribute or value until it reaches the leaf node. Complete the division of the sample.

The basic learner in the random forest is the CART algorithm. Classification and regression tree (CART) model is a commonly used decision tree method, which can be applied to both classification and regression problems. The CART algorithm is mainly composed of the following two steps:

(1) Decision tree generation

Use training samples to generate a decision tree. In general, the larger the tree, the better.

The generation of the decision tree is the process of continuously generating the binary tree from top to bottom. For how to generate a binary tree, that is, how to select feature, for the classification decision tree, the Gini index minimization criterion is used.

In the classification problem, assuming a total of K classes, the probability that an object can be classified into the k_th class is P_k, then the Gini index is:

$$Gini(p) = \sum_{k}^{K} p_k(1 - p_k) = \sum_{k}^{K} p_k^2 \tag{23}$$

For binary classification problems:

$$Gini(p) = 2p(1 - p) \tag{24}$$

Suppose the sample set is D. On the division of a certain feature A, the sample can be divided into two parts: D_1 and D_2, then the Gini index of the sample is:

$$Gini(D, A) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2) \tag{25}$$

When dividing the attributes, choose the category with the smaller Gini index to divide. Because the Gini index represents an uncertainty, the smaller the Gini index, the smaller the uncertainty, the more accurate the classification of categories.

(2) Decision tree pruning

Pruning the generated tree according to the verification data set and selecting the optimal subtree to avoid overfitting.

Decision tree pruning is essentially a simplified process of the generated tree in order to improve the ability to predict unknown data and avoid the occurrence of overfitting. The pruning of the

decision tree is divided into two steps. The first step is to start pruning from the bottom of the generated CART tree until the root node, in this process a series of subtree sequences are generated; the second step is to test the subtree sequence to select the optimal subtree, the method is to perform cross-validation on an independent verification data set.

4.3 Bagging and Random Forest

According to the above introduction, random forest is a strong learner built by Bagging integration method based on decision tree. Random forest can be used to solve classification problems, regression problems or other problems.

Bagging is a typical representative of parallel integration methods. It resamples by bootstrap method. Given a data set containing m samples, randomly select one from the sample data set and put it into the sampler, and then put the sample data back into the original data set. At the time of extraction, the data may be selected. After m times of extraction with replacement, a sample set containing m samples is obtained. The result of this kind of sampling is that some data are drawn multiple times and some data are not drawn once. Research show that the probability of each sample being drawn by this resampling technique is 63.2%.

According to the above method, extract T sample sets containing m samples. For each sample set, train a base learner, and then combine these base learners for output. This is the basic flow of the Bagging integration method. For classification problems, the combination of basic learners is the simple voting method. For regression problems, the combination of basic learners is the simple average method.

Random forest is an algorithm based on Bagging integration. Integrate the base learner with the Bagging method, and the base learner used in random forest is adecision tree. But in addition to this, the choice of random attributes is added to the base decision tree in the random forest when selecting attribute divisions. Generally, when choosing attribute division, the decision tree selects an optimal attribute among all attributes of the current node (assuming d attributes), and the base decision tree in the random forest randomly selects k attributes from the current node's attribute set for division when selecting attribute divisions. The parameter k determines the degree of introduction of the random attributes. When $k = d$, the base decision tree in the random forest is the same as the ordinary decision tree, When $k = 1$, an attribute is randomly selected for division. In general,$k = log_2^d$.

The random forest algorithm is relatively simple, has low computational overhead, and is easy to implement. A lot of research and practice show that the random forest algorithm can solve the prediction problem well, can avoid overfitting, and has a strong generalization ability. Even for some small sample data sets with missing data, it also has good prediction ability[17,18].

5 Markov Chain Principle

Russian mathematician Markov (Markov) based on Chebyshev's research on the limit law in probability theory, studied independent random variables and classical extreme value theory, improved the law of large numbers and the central extreme value theory. In the process of Markov's research on random variable sequences, a Markov stochastic process, or "Markov process", was proposed. On this basis, Markov chain theory came into being.

(1) Markov process

Markov process refers to: when the state of a system or process at the current time t_0 is known, the state at the next time t_1 ($t_1 > t_0$) is only related to the current time t_0, and it has nothing to do with other time before time t_0. That is to say, the system or process state value at the next moment in the future is only related to the current moment, but not to the historical moment. This process is called Markov process, and it refers to the random process of time transition and state transition.

(2) Markov chain principle

Markov chain is a special case of Markov process, that is, time and state variables are discrete and their states are finite and countable.The core idea of Markov method is to decide the state transferring probability matrix.[19,20] The transition probability of state E_i to state E_j after k times is determined as follows.

$$P_{ij}^{(k)} = \frac{m_{ij}^{(k)}}{M_i} \tag{26}$$

where M_i indicates the total number of states, $m_{ij}^{(k)}$ is the number of times the state E_i transferred to the state E_j. Then the $n \times n$ transfer matrix $P^{(k)}$ is expressed as follows.

$$p^{(k)} = \begin{bmatrix} p_{11}^{(k)} & p_{12}^{(k)} & p_{13}^{(k)} & \cdots & p_{1n}^{(k)} \\ p_{21}^{(k)} & p_{22}^{(k)} & p_{23}^{(k)} & \cdots & p_{2n}^{(k)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{m1}^{(k)} & p_{m2}^{(k)} & p_{m3}^{(k)} & \cdots & p_{mn}^{(k)} \end{bmatrix} \tag{27}$$

The state transition probability matrix is used to correct the original data by Markov chain prediction model.

$$X^k = X^0 P^k \tag{28}$$

where X^k is the probability transition matrix in time k, and X^0 is the probability vector of the initial state. The prediction results are corrected according to the following formula.

$$F = \frac{F_g}{1-q} \tag{29}$$

where F_g is the prediction value, and q is the boundary value of the original state interval.

Notes and references

1 J. Deng, *Syst. Control. Lett.*, 1982, **1**, 288–294.

2 E. Granada, J. Morán, J. L. Míguez and J. Porteiro, *Fuel. Process. Technol.*, 2006, **87**, 123–127.

3 Y. Kuo, T. Yang and G.-W. Huang, *Comput. Ind. Eng.*, 2008, **55**, 80–93.

4 J. Kennedy and R. Eberhart, *Proceedings of IEEE International Conference on Neural Networks*, 1995, **4**, 1942–1948.

5 Y. Shi and R. C. Eberhart, *Proceedings of the 1999 Congress on Evolutionary Computation*, 1999, **3**, 1945–1950.

6 L. D. S. Coelho and C. A. Sierakowski, *Adv. Eng. Softw.*, 2008, **39**, 877–887.

7 S. Suresh, P. B. Sujit and A. K. Rao, *Compos. Struct.*, 2007, **81**, 598–605.

8 K. E. Parsopoulos and M. N. Vrahatis, *Nat. Comput.*, 2002, **1**, 235–306.

9 C. Cortes and V. Vapnik, *Mach. Learn.*, 1995, **20**, 273–297.

10 J. Zhang, J. Wu, L. Tang, J. Jiang, G. Shen and R. Yu, *Anal. Methods*, 2015, **7**, 5108–5113.

11 J. Li, C. Zhao, W. Huang, C. Zhang and Y. Peng, *Anal. Methods*, 2014, **6**, 2170–2180.

12 X. Wang, D. Luo, X. Zhao and Z. Sun, *Energy*, 2018, **152**, 539–548.

13 A. Ben-Hur and J. Weston, *Methods. Mol. Biol.*, 2010, **609**, 223–239.

14 J. R. Quinlan, *Mach. Learn.*, 1986, **1**, 81–106.

15 J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

16 L. Breiman and etc, *Classification and Regression Trees*, Wadsworth Publisher, 1984.

17 L. Breiman, *Mach. Learn.*, 2001, **45**, 5–32.

18 L. R. Iverson, A. M. Prasad, S. N. Matthews and M. Peters, *Forest. Ecol. Manag.*, 2008, **254**, 390–406.

19 E. Hatzipantelis, A. Murray and J. Penman, *Artificial Neural Networks*, 1995, **409**, 369–374.

20 A. Dahamsheh and H. Aksoy, *Arab. J. Sci. Eng.*, 2014, **39**, 2513–2524.