Multiscale and Multidisciplinary Modeling, Experiments and Design (2024) 7:543–555
https://doi.org/10.1007/s41939-023-00225-1

ORIGINAL PAPER

# An explanatory machine learning model for forecasting compressive strength of high-performance concrete

Guifeng Yan[1] · Xu Wu[1] · Wei Zhang[1] · Yuping Bao[1]

Received: 18 April 2023 / Accepted: 10 August 2023 / Published online: 7 September 2023

**Abstract**
High-performance concrete (HPC) is one of the concrete types with high strength, good performance, and high durability, which has been considered in the structural industry. Testing and sampling this type of concrete to determine its mechanical properties is daunting and complex. In addition, human and environmental factors were significant in the preparation of samples, which was also time-consuming and energy-consuming. Artificial intelligence (AI) can be used to eliminate and reduce these factors. This article intends to use the machine learning (ML) method to forecast one of the HPC mechanical properties: compressive strength (CS). The experimental data set used from the published article includes 168 samples, of which 70% (118) of the sample belonged to training and 30% (50) to the testing phase. Least square support vector regression (LSSVR) is one of the ML models used for forecasting in this article. In addition, meta-heuristic algorithms have been utilized to obtain the target to improve the accuracy and reduce the error. Algorithms include Honey Badger algorithm (HBA), COOT optimization algorithm (COA), and generalized normal distribution optimization (GNDO). Combining the algorithms with the introduced model forms a hybrid format evaluated by metrics in the training and testing phases. By evaluating the hybrid models, it has been determined that they can forecast with high accuracy and are reliable. In general, the LSHB hybrid model obtained the highest $R^2$ and the lowest error compared to other models.

**Keywords** High-performance concrete · Compressive strength · Least square support vector regression · Honey Badger algorithm · COOT optimization algorithm · Generalized normal distribution optimization

## 1 Introduction

High-performance concrete (HPC) is the material of a building defined as concrete that satisfies high durability, strength, and workability (Aïtcin 1998). The American Concrete Institute (ACI) identifies HPC with specific combinations of uniformity and performance needs that are not always routinely achievable, operating normal mixing, established ingredients, placement, and curing practices (Russell 1999). While normal concrete combines water, coarse and fine aggregates, and Portland cement, HPC uses silica fume, fly ash, and superplasticizer as mineral and chemical admixtures (Bharatkumar et al. 2001; Lim et al. 2004). Using mineral admixtures as partial cement substitutes increases concrete behaviour via acting as pozzolanic materials and mineral admixtures. Conversely, chemical admixtures improve the HPC compressive strength (CS) by decreasing the porosity and water content within the hydrated cement paste (Larrard and Malier 2018; Huang et al. 2022).

Compressive strength (CS) is an important mechanical property, perhaps the most important quality of concrete, and is usually achieved by measuring concrete samples after standard curing for 28 days (Ni and Wang 2000). Therefore, developing forecasting models for the early specification of CS has received much attention. CS values can be forecasted using nonlinear and linear regression models for conventional concrete. However, for HPC, the relationship between input factors and CS is highly nonlinear; it gets complicated as the number of input factors enhancements. Therefore, regression models are improper for HPC's estimation of CS values. Hence, models according to artificial intelligence (AI) are attracting attention (Benemaran and Esmaeili-Falak 2020; Sarkhani Benemaran et al. 2022; Yin et al. 2021; Cheng et al. 2022). Forecasting the nonlinear relationship between concrete composition and strength based

✉ Xu Wu
Wuxu7109@163.com

1 Department of BIM Research, Nantong Institute of Technology, Nantong 226002, Jiangsu, China

on machine learning (ML) demands complete, large, consistent data sets. However, data sets are often incomplete due to missing data corresponding to various input characteristics, making it difficult to develop robust ML-based forecast models, so the availability of such data sets is critical. Moreover, as the complexity of these ML models improves, the results become more difficult to interpret. Interpreting these results is important for developing efficient material design processes to improve material performance (Lyngdoh et al. 2022; Yaltaghian and Khiabani 2023; Masoumi et al. 2020).

Yeh (1998) suggested a new neural network architecture and studied its efficiency and precision in modelling concrete CS. Sobhani et al. (2010) investigated some adaptive neuro-fuzzy inference systems (ANFIS), artificial neural networks (ANN), and regression models for estimating 28-day CS of concrete. They indicated that ANFIS and ANN could estimate CS with satisfactory performance, but regression models are unreliable. Prasad et al. (Prasad et al. 2009) employed ANN for forecasting 28-day CS of normal, HPC, and high-strength self-compacting concrete (SCC) with large amounts of fly ash. Alshihri et al. (Alshihri et al. 2009) studied a method to forecast the CS of lightweight concrete (LWC) utilizing backpropagation (BP) and cascade correlation (CC) neural networks. Their identification demonstrated that the ANN model was sufficient to forecast the LWC CS.

Topcu and Saridemir (2008) investigated a fuzzy logic model with ANN to forecast 7-, 28-, and 90-day concrete CS, including low- and high-lime fly ash. Their conclusions indicated that fuzzy logic models and ANN are practical models to forecast concrete CS. Chen et al. (2012) suggested a hybrid for estimating CS of HPC by incorporating fast, chaotic genetic algorithms in an evolutionary fuzzy support vector machine inference model (EFSIMT), weighted support vector machines (SVM), and fuzzy logic for time series data. They showed that the EFSIMT approach obtained higher performance goals compared to SVM. In addition, they determined that SVM performed the worst compared to ANN and EFSIMT. Zhang et al. (2020a) used the random forest (RF) algorithm to calculate variable importance in their study, and they imputed the data using kNN-10. In a separate study, Nguyen et al. (2021) used XGBoost to generate feature importance for concrete tensile strength. Both studies assessed feature importance by directly examining the data or observing the decrease in model performance. However, there has been no effort to integrate data interpretation algorithms with machine learning (ML)-based models. Therefore, there is a need for a more efficient integration of a robust data interpretation algorithm with ML-based models for concrete strength prediction, to develop a powerful and interpretable predictive tool.

The novelty of this article is to enhance the SVM model's accuracy in estimating the HPC CS. Despite the many advantages of SVMs, the theory of SVMs only covers parameter specification and kernel selection for specific values of regularization and kernel parameters. In addition, the existing SVM method has a very complex algorithm and requires much memory. This study proposed the least square support vector regression (LSSVR) model, a lower calculation cost conversion of the support vector regression (SVR) model. Nevertheless, like SVR, the LSSVR's effectiveness model relies on kernel parameter settings and proper regularization, which can be defined as optimization issues.

Moreover, the meta-heuristic algorithm has been combined with LSSVR to improve the efficiency and accuracy of forecasting CS and optimize the output. The algorithms are included Honey Badger algorithm (HBA), COOT optimization algorithm (COA), and generalized normal distribution optimization (GNDO). The models were in hybrid form and framework of LSSVR + HBA (LSHB), LSSVR + COA (LSCO), and LSSVR + GNDO (LSGN). The hybrid models have been evaluated with several indicators for choosing the most suitable model. In addition, the models' performance compared via the experimental data set used from the published article include 168 samples, of which 70% (118) of the sample belonged to training and 30% (50) to the testing phase.

## 2 Materials and methodology

### 2.1 Data gathering

This study used the experimental data set published in Lam et al. (1998). The main concrete variables are water, coarse and fine aggregates and binders. In addition to these main ingredients, other additives, such as fly ash (FA), micro-silica (MS), and superplasticizer (SP), can be added to the mixture to improve concrete properties. On the other side, it is clear that the longer the time of the curing, the higher the CS. Hence, considering to use all these components as inputs for improving the completeness of the model. Eight components were chosen as input components based on their impact on CS forecasting using FA and MS in samples of different ages. The input variables consisted of sample age (age), superplasticizer to total binder ratio (SP/B), micro-silica to total binder ratio (MS/B), fly ash to total binder ratio (FA/B), water to total binder ratio (W/B), the ratio of coarse aggregate to total binder (CA/B), coarse aggregate to the total aggregate ratio (CA/TA), and total binder content (B), which the output variable is CS. Tables 1 and 2 indicate 70% (118) of training and 30% (50) of testing samples, respectively. Furthermore, the minimum (min), standard deviation (St. dev.), maximum (max), and average (avg) values of variables.

**Table 1** Statistical attributes of inputs and CS in the train phase

| Indicators | Variables | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Input | | | | | | | | Targets |
| | $B$ (kg/m$^3$) | FA/B % | MS/B % | CA/B % | CA/TA% | SP/B% | W/B % | Age (day) | CS (MPa) |
| Min | 394 | 0 | 0 | 2.172 | 0.6 | 0.3 | 0 | 28 | 24 |
| Max | 500 | 0.55 | 0.110 | 2.906 | 0.679 | 0.5 | 2.6 | 180 | 107.8 |
| Avg | 429.62 | 0.247 | 0.032 | 2.658 | 0.634 | 0.410 | 1.05 | 78.50 | 65.09 |
| St. dev | 44.23 | 0.176 | 0.038 | 0.308 | 0.022 | 0.081 | 0.755 | 57.14 | 17.59 |

**Table 2** Statistical attributes of inputs and CS in the test phase

| Indicators | Variables | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Input | | | | | | | | Targets |
| | $B$ (kg/m$^3$) | FA/B % | MS/B % | CA/B % | CA/TA% | SP/B% | W/B % | Age (day) | CS (MPa) |
| Min | 394 | 0.15 | 0.090 | 2.906 | 0.615 | 0.3 | 1 | 28 | 46.5 |
| Max | 394 | 0.3 | 0.110 | 2.906 | 0.615 | 0.5 | 1 | 56 | 86.43 |
| Avg | 394 | 0.227 | 0.102 | 2.906 | 0.615 | 0.377 | 1 | 47.02 | 61.73 |
| St. dev | 0 | 0.053 | 0.008 | 4.5E-16 | 5.6E-16 | 0.080 | 0 | 13.19 | 8.42 |

## 2.2 Least square support vector regression (LSSVR)

The least squares support vector machine algorithm (LS-SVM) is a refinement of the standard SVM, which is computationally more computationally efficient than the standard SVM by transforming the quadratic optimization issues into a transformed, more linear system of equations. Instead of solving the quadratic programming issues in standard SVM, the LS-SVM algorithm solves a linear set of equations. LS-SVM is able to be utilized for both regression and classification troubles. The present work uses LS-SVR to predict the CS of HPC in regression form. Here is an overview of LS-SVR:

Input $x_i$ (experimental variable) and output $y_i$ (forecasted value: local precipitation). Based on the LS-SVR approach, the nonlinear LS-SVR function is able to be presented in the following equation:

$$f(x) = w^{'} m + t \tag{1}$$

here $f$ represents the relationship between local precipitation and experimental variables, and $w$ is m-dimensional weight vectors, $m$ and $t$ show mapping functions and bias terms, alternatively (Kisi 2015).

The regression issue employing the function forecasting error for the structure minimization principle can be presented in the following:

$$\min G(w, r) = \frac{1}{2} w^{'} w + \frac{p}{2} \sum_{i=1}^{n} r_i^2 \tag{2}$$

It has the following limitations:

$$y_i = w^{'} m(x_i) + t + r_i, \quad i = 1, 2, \ldots, n \tag{3}$$

where $r_i$ represents the training error of $x_i$ and p is the penalty term.

For solving for $w$ and $r$, use Lagrange multiplier optimal programming for solving Eq. (2). The objective function can be defined by transforming the constrained issue into an unconstrained one. The Lagrange function F is able to be determined in the following equation:

$$F(w, t, r, a) = G(w, r) - \sum_{i=1}^{n} a_i \left\{ w^{'} m(x_i) + t + r_i - y_i \right\} \tag{4}$$

Here $a_i$ is the multipliers' Lagrange. Considering the Karush–Kuhn–Tucker (KKT) condition, the partial results of Eq. (5) for $w$, $t$, $r$, and $a$, alternatively are

$$\begin{cases} w = \sum_{i=1}^{n} a_i m(x_i) \\ \sum_{i=1}^{n} a_i = 0 \\ a_i = pr_i \\ w' m(x_i) + t + r_i - y_i = 0 \end{cases} \tag{5}$$

Then the linear equation can be derived after removing $r_i$ and $w$ in the following equation:

$$\begin{bmatrix} 0 & Y' \\ Y & ZZ' + \frac{H}{p} \end{bmatrix} \begin{bmatrix} t \\ a \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{6}$$

here $Y = (y_1, \ldots, y_n)$, $Z = \left(m(x_1)' y_1, \ldots, m(x_n)' y_n\right)$, $H = (1, \ldots, n)$, $a = (a_1, \ldots, a_n)$.

The function of kernel $C(x, x_i) = m(x_1)' m(x_i)$, $i = 1, \ldots, n$, satisfied with the condition of Mercer (Suykens et al. 2002). Consequently, LS-SVR can be expressed in the following equation:

$$f(x) = \sum_{i=1}^{n} a_i C(x, x_i) + t \tag{7}$$

## 2.3 Honey Badger algorithm (HBA)

The Honey Badger algorithm (HBA) mimics the Honey Badgers searching manner (Hashim et al. 2022). To find food sources, honeyguide birds are sniffed and dug or followed by honey badgers. The first case is named digging mode, and the second is named honey mode. In the previous mode, it utilizes scent capability to get closer to the location of prey. Once there, it moves around its prey to choose a suitable spot to dig and capture it. In the latter mode, honey badgers use honeyguide bird guides for locating hives directly.

Initialize each location based on the badgers' number ($N$) and Eq. (8):

$$x_i = lb_i + r_1 \times (ub_i - lb_i) \tag{8}$$

where $r_1$ shows a random number between 0 and 1, $x_i$ shows the $i$th honey badger's position associated with the candidate solution for the $N$ population, and $ub_i$ and $lb_i$ show the explore region's upper and lower bounds, respectively.

The intensity was the distance between the honey badger and the prey and prey concentration. $L_i$ shows the prey odour severity. When the smell is powerful, the movement is fast and vice versa, represented by the inverse square law (Kapner et al. 2007) as given by the following equation:

$$\begin{aligned} L_i &= r_2 \times \frac{C}{4\pi d_i^2} \\ C &= (x_i - x_{i+1})^2 \\ d_i &= x_{\text{prey}} - x_i \end{aligned} \tag{9}$$

Here $r_2$ is the random number between 0 and 1, $C$ shows the strength of concentration, $d_i$ is the distance between the $i$th badger and the prey.

The factor of density ($f$) manages the time-varying randomness to provide a smooth transition from exploration to exploitation. In addition, updating the factor of the density ($d$), which reduces with repetition, to reduce the randomness over time utilizing the following equation:

$$a = S \times exp\left(\frac{-t}{t_{\max}}\right) \tag{10}$$

where $t_{\max}$ is a maximum iteration number, and $S$ shows a constant $\geq 1$ (default $= 2$).

Escaping from the local optimum step and the position of agents' phases are employed for exiting the optimal local region. The presented algorithm operates an indicator ($I$) that alterations the explore direction to take advantage of large opportunities for agents to traverse the search space closely.

As previously mentioned, the HBA ($x_{\text{new}}$) location update strategies are split into two parts, the "honey phase" and the "digging phase". Here is a suitable illustration.

A badger shows an action resembling the Cardioid shape (Akopyan 2015) during the digging phase. Cardioid motion can be simulated by the following equation:

$$\begin{aligned} x_{\text{new}} &= x_{\text{prey}} + I \times b \times L \times x_{\text{prey}} + I \times r_3 \times a \\ &\quad \times d_i \times |\cos(2\pi r_4) \times [1 - \cos(2\pi r_5)]| \end{aligned} \tag{11}$$

here $x_{\text{prey}}$ shows the position of the prey; this is the finest position found thus far—that is to say, the finest overall position. $b \geq 1$ (default $= 6$) is the capability of the badger to get food, $r_3$, $r_4$, and $r_5$ shows 3 various accidental numbers among 0 and 1, and $I$ acts as a explore direction change flag, which is defined by the following equation:

$$\text{I} = \begin{cases} 1\, if & r_6 \leq 0.5 \\ -1 & else \end{cases} \tag{12}$$

A honey badger is highly dependent on the odour intensity of its prey, the distance between the badger and the prey $d_i$ and the time-varying foraging influence factor $a$ during the digging phase. In addition, a badger can pick up on any F-noise, allowing it to find its prey's location even better during foraging.

If the honey badger follows the honey leader to attain the hive, it is given by the following equation:

$$x_{\text{new}} = x_{\text{prey}} + I \times r_7 \times a \times d_i \tag{13}$$

where $x_{\text{new}}$ refers to the new position of the honey badger and $x_{prey}$ shows the position of the prey. $I$ and $a$ indicate calculated employing Eqs. (12) and (10), alternatively. Based on the distance information $d_i$, can observe how the badger explores near the prey position $x_{prey}$ found so far in Eq. (13). The exploration is subject to time-varying search behaviour. Honey badgers can also find $I$ obstacles. In addition, Algorithm 1 has shown the pseudo-code of HBA (Hashim et al. 2022).

## 2.4 COOT optimization algorithm (COA)

Coots are small waterfowl belonging to the Rallidae family. They form the genus Fulica, Latin for "coot" (Paillisson and Marion 2001).

The algorithm starts with $x = \{x_1, x_2,\ldots, x_n\}$ primary random populations (Naruei and Keynia 2021). For determining target values $V = \{V_1, V_2, \ldots, V_n\}$, the random population is frequently assessed by the target function. Furthermore, this is powered by a rules' set that form the optimization method's core. Population-based optimization methods search an optimal set of optimization issues, so it is not guaranteed that a solution will be found in a single pass. Thus, using sufficient optimization steps and random solutions increases the likelihood of finding the global optimum. The population is accidentally developed in visible space according to the following equation:

**Algorithm 1.** Honey Badger Algorithm

```
Set N, $t_{max}$, b, S parameters.
Initialize the population with random positions.
Asses and assign the goodness of fit of each honey badger location $x_i$ utilizing the objective function
$f_i, i \in [1, 2, \ldots, N]$
Store the finest $x_{prey}$ position and assign a fitness
while $t \leq t_{max}$ do
    Update the reducing factor utilizing Eq. (10).
    for $i = 1$ to $N$, do
        Compute the intensity by Eq. (9)
        if $r < 0.5$ then
            Update the $x_{new}$ by Eq. (11)
        else
            Update the $x_{new}$ by Eq. (13)
        end if
        Assess new position and assign to new fitness
        if $new\ fitness \leq f_i$ then
            Set $x_i = x_{new}$ and $f_i = new\ fitness$
        end if
    end for
end while
Stop criteria met.
```

$$P(i) = r(1, d) \times (ub - lb) + lb \tag{14}$$

here $P(i)$ shows the Coot's position, $lb$, $ub$ are the lower and upper bound of exploring space, $d$ is the dimensions of the trouble.

In addition, we need to compute each solution's fitness utilizing the objective function $\text{Oi} = f(x)$ after specifying the location of each agent and producing the initial population. Select several coots as the group leader. Leader selection is random.

We have implemented the four motions of the Coot on the water surface described in the earlier section.

To assume a random location based on Eq. (15) in the explore space and move the Coot toward the accidental location for achieving random motion.

$$G = r(1, d) \times (ub - lb) + lb \tag{15}$$

The Coot's movements explore various parts of the explore space. This move will bring the algorithm out of the local optimum if the algorithm is blocked in a local optimum. The new position of Coot is calculated on the basis of the following equation:

$$P(i) = P(i) + E \times r \times (G - P(i)) \tag{16}$$

where $r$ demonstrates a random number among 0 and 1, and $E$ is determined in the following equation:

$$E = 1 - T \times \left(\frac{1}{\text{Max}_{\text{Iter}}}\right) \tag{17}$$

where $\text{Max}_{\text{Iter}}$ shows the maximum iteration, and $T$ show the present iteration.

Chain movement can be implemented using the two coots' average position. Another way to achieve chain motion is to compute the distance vector among the two coots, then move the Great Coot closer to the other by about half the vector distance. Using the first method, Coot's new position is computed based on the following equation:

$$P(i) = 0.5 \times (P(i-1) + P(i)) \tag{18}$$

here $P(i-1)$ shows the second Coot.

Sometimes, the remaining coots should control their location according to the group's leader and approach them, and a few coots in the face of the group manage the group. The idea is to control its location according to the leader. The leader's average location can be considered, and the Coot can update its location according to this average location. Assuming the average location leads to premature convergence. For implementing the move, a can be chosen a leader using the mechanism based on the following equation:

$$L = 1 + (nN) \tag{19}$$

where L indicates the index number of the leader, $n$ shows the present coot number, and $N$ shows the number of leaders.

According to the leader's location, the Coot must update its position. The Coot's next location according to chosen leader can be computed in the following equation:

$$P(i) = l + 2 \times r \times \cos(2\pi r_1) \times (l - P(i)) \tag{20}$$

here $P(i)$ shows the Coot's present location, $l$ shows the location of chosen leader, and $r_1$ indicates a random number in the interval [− 1, 1].

The group needs to align itself toward the purpose, so the leader needs to update his position toward the goals. Equation (21) recommends updating the leader's location as the equation searches the suitable locations around the present suite spot. Leaders must pull away from their present optimal location to find a suitable position. The equation delivers a great way to get away from or near the optimal location.

$$l = \left\{\begin{array}{ll} D \times r \times \cos(2\pi r_1) \times (B - l) & if \quad r < 0.5(a) \\ D \times r \times \cos(2\pi r_1) \times (B - l) - B & if \quad r \geq 0.5(b) \end{array}\right\} \tag{21}$$

Here $B$ shows the finest location found so far, and $D$ is computed based on the following equation:

$$D = 1 - T \times \left(\frac{1}{\text{Max}_{\text{Iter}}}\right) \tag{22}$$

Moreover, the COA pseudo-code has been indicated in Algorithm 2.

**Algorithm 2.** COOT Optimization Algorithm

```
Initialize the coots' first population randomly by Eq. (14)
Initialize the parameters of N with random positions.
Compute the fitness of leaders and coots
Calculate E and D parameters based on Eq. (17) and Eq. (22)
if $r < 0.5$ do
    for $i = 1$ to the coots' number, do
        Compute the L by Eq. (19)
        if $r > 0.5$ then
            Update the coots' location by Eq. (20)
        else
        if $r < 0.5$ then
            Update the coots' location by Eq. (18)
        else
            Update the coots' location by Eq. (16)
        end if
    end for
    Calculate Coot's fitness
        if $the\ coot's\ fitness \leq the\ leader\ coot$ then
    For leaders' number
            if r < 0.5
            Update the leader's location by Eq. (21)(a)
        else
            Update the leader's location by Eq. (21)(b)
        end if
    end for
Stop criteria met.
```

## 2.5 Generalized Normal distribution optimization (GNDO)

Generalized normal distribution optimization (GNDO) mimics the normal distribution theory (Zhang et al. 2020b). The normal distribution is a very important tool for explaining natural phenomena called Gaussian distribution. The normal distribution can be determined by assuming a random variable x follows a contingency distribution with position parameter μ and scale parameter δ, whose contingency density function is able to be stated as

$$f(x) = \frac{1}{\sqrt{2\pi\delta}}.\exp\left(-\frac{(x-\mu)^2}{2\delta}\right) \tag{23}$$

where $x$ shows a normal random variable and the distribution is normal, μ is the location parameter, and δ is the scale parameter used to determine the mean and standard variance of the random variable, respectively. Local exploitation refers to finding the finest solution in the exploration space that contains everyone's present location. Can construct a generalized normal distribution model for optimization based on the relationship between the normal distribution and the distribution of individuals in the population in the following equation:

$$v_i^t = \mu_i + \delta_i \times \eta,\ i = 1, 2, 3, \ldots, N \tag{24}$$

where $v_i^t$ denotes the $i$th individuals' tracking vector at time $t$, $\mu_i$ denotes the $i$th individual's generalized mean position, $\delta_i$ denotes the generalized standard variance, and $\eta$ is the penalty that shows the coefficient. Furthermore, $\mu_i$, $\delta_i$, and $\eta$ can be determined in the following:

$$\mu_i = \frac{1}{3}(x_i^t + x_{\text{best}}^t + M) \tag{25}$$

$$\delta_i = \sqrt{\frac{1}{3}[(x_i^t - \mu)^2 + (x_{\text{best}}^t - \mu)^2 + (M - \mu)^2]} \tag{26}$$

$$\eta = \begin{cases} \sqrt{-\log(\lambda_1)} \times \cos(2\pi\lambda_2), & if \quad a \le b \\ \sqrt{-\log(\lambda_1)} \times \cos(2\pi\lambda_2 + \pi), & if \quad a > b \end{cases} \tag{27}$$

where $a$, $b$, $\lambda_1$, and $\lambda_2$ denote random numbers between 0 and 1, $x_{\text{best}}^t$ denotes the present finest position, and $M$ denotes the present population means position. In addition, M can be computed in the following equation:

$$M = \frac{\sum_{i=1}^{N} x_i^t}{N} \tag{28}$$

Global exploration is searching language zones around the world to find hopeful zones. The global scan on GNDO assumes 3 accidentally elected persons, which can be expressed as

$$v_i^t = x_i^t + \beta \times (|\lambda_3| \times v_1) + (1 - \beta) \times (|\lambda_4| \times v_2) \tag{29}$$

where $\lambda_3$ and $\lambda_4$ denote two random numbers following a standard normal distribution, $\beta$ is a tuning parameter denoting a random number between 0 and 1, and $v_1$ and $v_2$ denote two tracking vectors. In addition, $v_1$ and $v_2$ can be computed in the following:

$$v_1 = \begin{cases} x_i^t - x_{p1}^t, & if \quad f(x_i^t) < f\left(x_{p1}^t\right) \\ x_{p1}^t - x_i^t, & otherwise \end{cases} \tag{30}$$

$$v_2 = \begin{cases} x_{p2}^t - x_{p3}^t, & if \quad f\left(x_{p2}^t\right) < f\left(x_{p3}^t\right) \\ x_{p3}^t - x_{p2}^t, & otherwise \end{cases} \tag{31}$$

where $p1$, $p2$, and $p3$ indicate 3 random integers chosen from 1 to $n$.

## 2.6 Performance evaluation methods

Five performance indicators are commonly employed in concrete estimative evaluations. In addition, they were used for assessing the machine learning method proposed in this paper. The correlation coefficient ($R^2$) determines an amount of how acceptable the explanatory variables explain the variable's measured response. Provides the virtue of fit of the model. The suitable estimative potential of such a model is defined with a higher $R^2$ value. Root mean squared error (RMSE) shows a forecast accuracy's statistical measure. RMSE calculates a response variable's variance, which can be described via models. Mean absolute error (MAE) indicates a statistic that computes the average error size of the model's forecasting. Furthermore, variance account factor (VAF) and mean absolute percentage error (MAPE) are presented in the following:

$$R^2 = \left( \frac{\sum_{i=1}^{n}(p_i - \overline{p})(r_i - \overline{r})}{\sqrt{\left[\sum_{i=1}^{n}(p_i - p)^2\right]\left[\sum_{i=1}^{n}(r_i - \overline{r})^2\right]}} \right)^2 \tag{32}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - p_i)^2} \tag{33}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|p_i - r_i| \tag{34}$$

$$\text{MAPE} = \frac{100}{n}\sum_{i}^{n}\frac{|t_i|}{|p_i|} \tag{35}$$

$$\text{VAF} = \left(1 - \frac{var(r_i - p_i)}{var(r_i)}\right) \times 100 \tag{36}$$

where $p_i$ and $r_i$ determine the forecasted and measured values, $\overline{r}$ and $\overline{p}$ show the mean values of measured and forecasted, respectively; also, $n$ is the sample number.

## 3 Results and discussion

### 3.1 Hyperparameter

LSSVR is a type of support vector machine (SVM) that is used for regression tasks. Like other machine learning algorithms, LSSVR has hyperparameters that must be set before training the model (Kovačević et al. 2016, 2021). Table 3 shows the important hyperparameters of LSSVR. For each optimization algorithm, the table shows the values of C and Gamma that were found to give the best performance on the data set being used. Specifically, for LSHB, the optimal hyperparameters were found to be $C = 938.505$ and Gamma $= 917.994$; for LSCO, the optimal hyperparameters were C = 895.8666 and Gamma = 801.9936; and for

**Table 3** Results of hyperparameters

| Models | Hyperparameter | |
|---|---|---|
| | *C* | Gama |
| LSHB | 938.5047 | 917.9939 |
| LSCO | 895.8666 | 801.9936 |
| LSGN | 980.1297 | 793.2765 |

**Table 4** Initiation parameter HBA

| Initiation parameter | Optimizer | | |
|---|---|---|---|
| | HBA | COA | GNDO |
| Population size | 50 | 50 | 50 |
| Beta | 6 | – | – |
| C | 2 | – | – |
| $\text{Max}_{\text{Iter}}$ | 100 | 100 | 100 |
| Lower bound | 0 | 0 | 0 |
| Higher bound | 1000 | 1000 | 1000 |
| n_leader | – | 5 | – |
| n_COOT | – | 45 | – |

LSGN, the optimal hyperparameters were $C = 980.1297$ and Gamma $= 793.2765$. These hyperparameters were likely found through a process of hyperparameter tuning, where the optimization algorithms were used to search through a range of possible hyperparameter values to find the combination that resulted in the best performance on a held-out validation set. The specific range of searched hyperparameter values and the performance metric used to evaluate the models are not shown in the table but would be important to consider for a full understanding of the hyperparameter tuning process.

In addition, Table 4 displays the initiation parameters for three optimization algorithms (HBA, COA, and GNDO) in the context of a practical implementation. The initiation parameters are used to set up the optimisation algorithms' initial conditions, including the population size, the range of variable values, and the maximum number of iterations. For HBA, the population size is 50, and the maximum number of iterations is 100. The beta and C parameters are also specified, with beta set to 6 and C set to 2. The lower and upper bounds are set to 0 and 1000, respectively. For COA, the population size is also set to 50, and the maximum number of iterations is set to 100. In addition, the number of leaders and COOTS are specified, with n_leader set to 5, and n_COOT set to 45. The lower and upper bounds are set to 0 and 1000, respectively. For GNDO, the population size and maximum number of iterations are as same as other optimizers. The lower and upper bounds are set to 0 and 1000, respectively. It is worth noting that these initiation parameters are specific to the problem being addressed and may need to be adjusted for different problems or optimization algorithms.

**Table 5** Results obtained from the proposed models

| | Phase | $R^2$ | RMSE | MAE | MAPE | VAF |
|---|---|---|---|---|---|---|
| Methods | | | | | | |
| LSHB | Train | 0.999 | 0.377 | 0.2342 | 0.4280 | 99.952 |
| | Test | 0.997 | 1.408 | 1.0551 | 1.6294 | 96.46 |
| LSCO | Train | 0.996 | 0.994 | 0.5727 | 1.112 | 99.67 |
| | Test | 0.988 | 2.214 | 1.6987 | 2.720 | 86.62 |
| LSGN | Train | 0.993 | 1.353 | 0.7793 | 1.514 | 99.38 |
| | Test | 0.982 | 2.579 | 1.9790 | 3.165 | 79.66 |

## 3.2 Comparing the performance

This section examines the present models according to the metrics explained in the previous section. In evaluating the models, the highest value of $R^2$ and VAF and the lowest value of RMSE, MAE, and MAPE are considered. Table 5 demonstrates the obtained model values in the evaluators in two testing and training phases. 70% and 30% of the samples are assigned to the training and testing phase. According to Table 5, the highest $\text{R}^2$ value equal to 0.999 was obtained for $\text{LSHB}_{\text{Train}}$, and the lowest value was obtained by $\text{LSGN}_{\text{Test}}$ equal to 0.982. For RMSE, the lowest value was for $\text{LSHB}_{\text{Train}} = 0.377$, and the highest was for $\text{LSGN}_{\text{Test}} = 2.579$. In MAE, $\text{LSHB}_{\text{Train}} = 0.234$ had the finest performance, while $\text{LSGN}_{\text{Test}} = 1.979$ had the worst performance. The noteworthy point in evaluating the models is that all the models have obtained the highest value in the training section, and their performance has weakened in the testing section, which indicates that the models are not well trained in the training section. In MAPE, the most appropriate value was obtained by $\text{LSGB}_{\text{Train}} = 0.428$, while the poorest value belonged to $\text{LSGN}_{\text{Test}} = 3.165$. Finally, for VAF, which, like $\text{R}^2$, is the highest value of the finest performance criterion, the most satisfactory value is $\text{LSHB}_{\text{Train}} = 99.95$, and the lowest value is obtained by $\text{LSGN}_{\text{Test}} = 79.66$. From the strongest to the weakest performance, LSHB, LSCO, and LSGN have generally performed. In addition, it was shown that combining HBA with LSSVR can provide a suitable model with high accuracy for forecasting.

Figure 1 shows the Scatter plot of forecasted and measured CS in two testing and training phases. The corresponding shape is determined based on two evaluators, $R^2$ and RMSE, which determine being in a series and dispersion or density, respectively. In addition, the center line component, which is based on $X = Y$ coordinates, and the linear fit component have

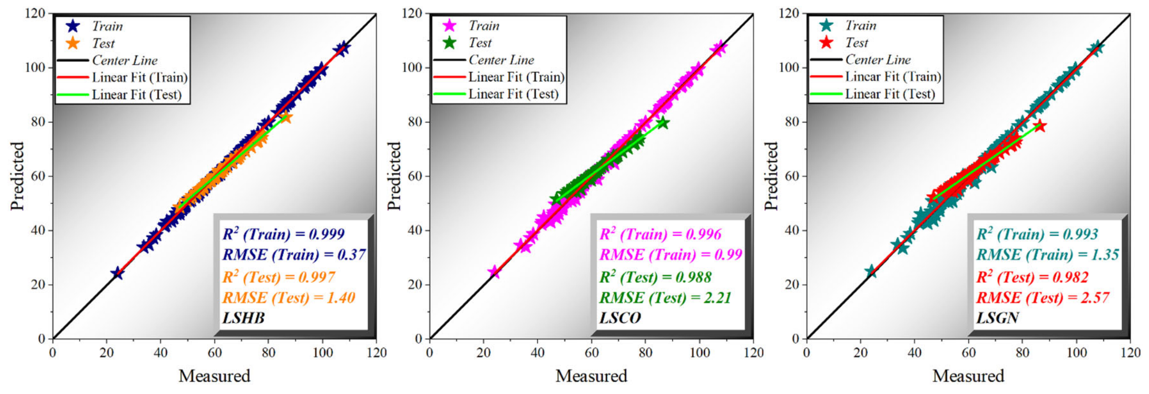
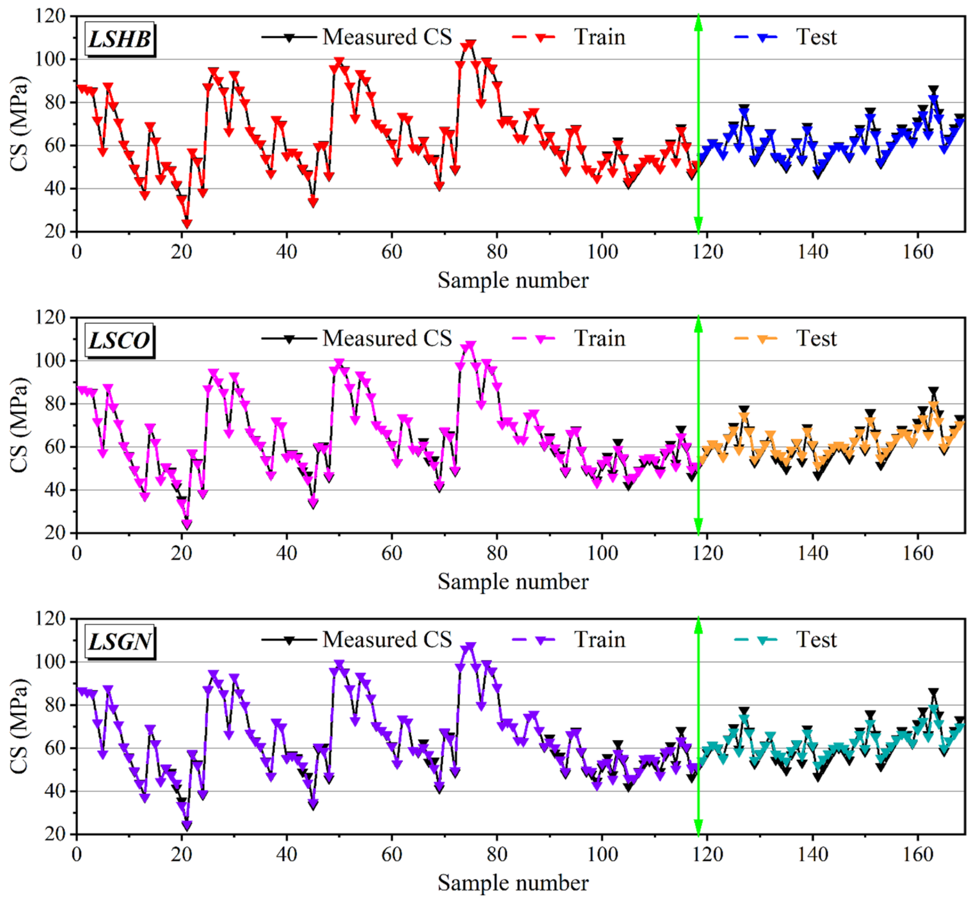


**Fig. 1** Scatter plot of forecasted and measured CS

**Fig. 2** Comparison between forecasted and measured CS



been in two phases of training and testing. The difference in the angle of these two lines indicates the appropriate or poor performance of the models. In LSHB, due to the high value of $R^2$ and the lower RMSE in training compared to testing; as a result, it can be seen that there is less dispersion in training compared to testing, and the difference between the angle of linear fit and the center line is smaller. On the other hand, LSCO and LSGN models have similar performance in the two phases of training and testing. However, considering that LSGN has the highest RMSE and the lowest $R^2$ in both the testing and training phases, it can be seen that the scatter of points in this model is more than the other two models.

Figure 2 compares forecasted and measured CS in two training and testing phases. The ideal situation is such that the forecasted and measured behaviour is similar. The models performed almost the same in the training phase, but with a closer look at samples 100–109, it can be seen that LSHB had a smaller difference with the measured than the other two models. In addition, it can be seen in the test phase that LSHB has the most appropriate performance, and LSGN has

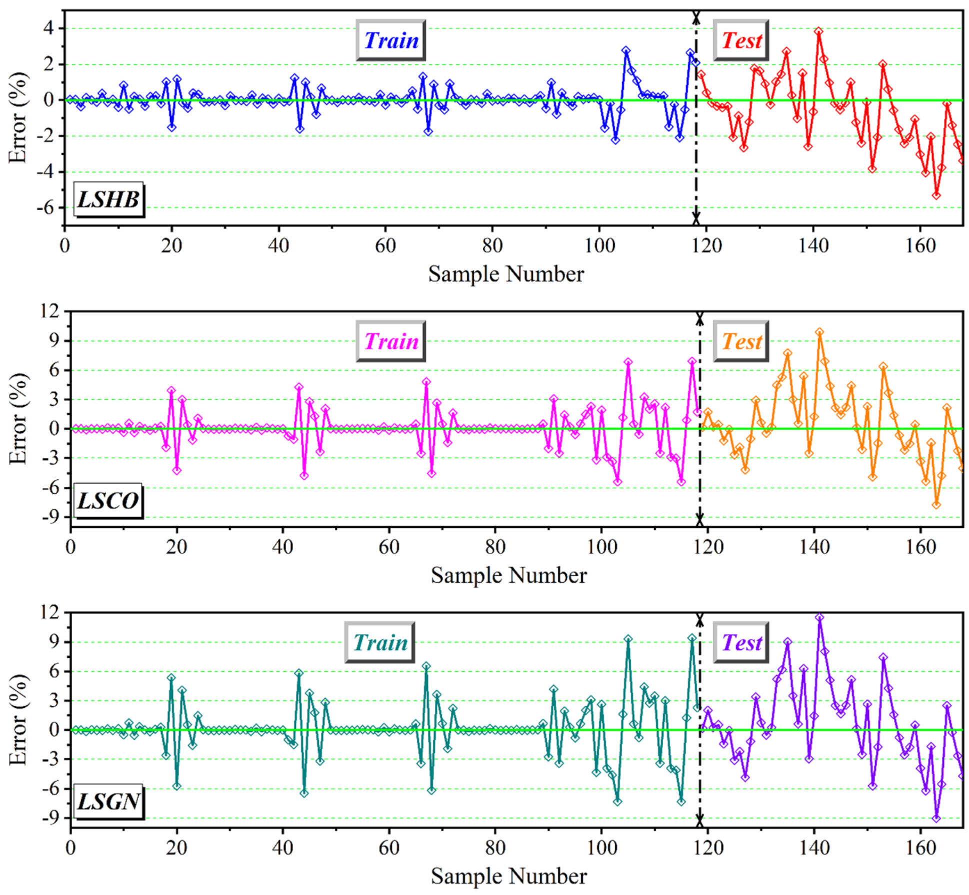


**Fig. 3** Error percentage in the train and test phase

shown a relatively weak performance. In general, it is able to be deduced that the models performed more satisfactorily in the training phase than the test, and the combined LSHB model had higher accuracy in both phases compared to the LSCO and LSGN models.

Figure 3 shows the error percentage in the train and test phases. As mentioned, 70% of the samples were related to the training section and 30% to the testing section. The LSHB model recorded the highest error percentage in the training at almost 3%, which increased to 5.5% as the performance weakened. In LSCO, the highest percentage of error in training was equal to 6.2%, which, like LSHB, was obtained by increasing the error in testing equal to 9.2%. Finally, for LSGN, the highest error percentage was equal to 9%, the highest among the models, and the error in the test reached almost 12%. In general, it can be seen that the LSHB model had the least error in the two phases of training and testing, and the other two models were also reliable for forecasting.

Figure 4 shows the box normal for the error percentage of the presented models. In LSHB, it can be seen that in the training phase, the average error was 0%, and its normal distribution was sharp and normal, showing this less dispersion. In addition, the errors were good in the dispersion test, below 10%. In LSCO, the dispersion of samples has been evident in

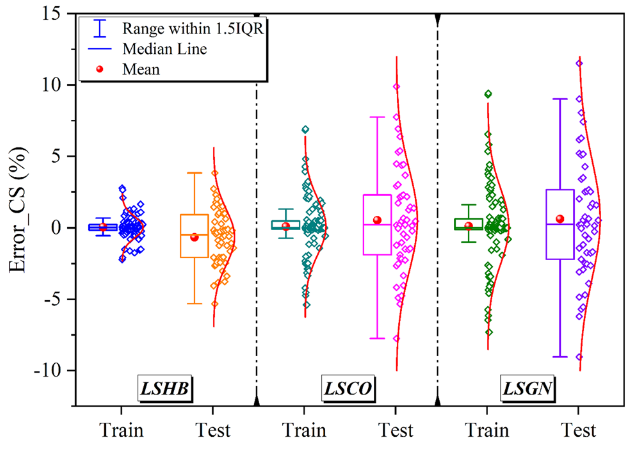


**Fig. 4** Box-normal for the error percentage of the presented models

both phases, and a flatter normal distribution has been registered, but despite this, the highest error below 10% has been obtained. Finally, for LSGN, which had the highest and most scattered error, only in the test phase was a sample above 10% obtained, which can be called outlier data. The normal distribution of LSGN is also flatter than the other two models and has a lower frequency close to zero. In general, it can be concluded that all three models performed well, but LSHB recorded the finest output.

## 4 Conclusion

Due to its durability and workability, high-performance concrete (HPC) has performed better than ordinary concrete in special structures. However, obtaining the values of the mechanical properties of this type of concrete in a laboratory has been a time-consuming and energy-consuming task. For this reason, learning machines have replaced human experiments. This study's novelty is introducing one of the machine learning methods, including LSSVR, to forecast HPC compressive strength (CS). In addition, to increase the accuracy and get the least errors, the corresponding model was combined with three meta-heuristic algorithms and formed a hybrid model, which the algorithms include Honey Badger algorithm (HBA), COOT optimization algorithm (COA), and generalized normal distribution optimization (GNDO). In addition, in this article, several metrics were used to evaluate the performance of the models, including $R^2$, RMSE, MAE, MAPE, and VAF. In evaluating the models, it was observed that HBA obtained a better combination with LSSVR. According to the obtained outputs, it was observed that in $\mathrm{R}^2$, the models obtained close results and did not differ much from each other. In RMSE, the most appropriate value obtained by LSHB with LSCO and LSGN has a difference of 62 and 72 per cent and 59 and 69 per cent recorded in MAE, respectively. The difference is obtained for MAPE with LSCO = 61.5% and LSGN = 71%. In VAF, the highest value obtained by LSHB in the training phase had values close to each other, but in the testing phase, the better performance of the models with each other was evident. In general, it can be concluded that machine learning methods are reliable for forecasting, and algorithms can increase the performance of models in terms of accuracy and error reduction.

**Author contributions** GY: methodology, formal analysis, software, validation. XW: writing—original draft preparation, conceptualization, supervision, project administration. WZ: software, formal analysis, methodology, language review. YB: writing—original draft preparation, methodology, software, language review.

**Funding** This work was supported by the Development and Application of an Intelligent Park Operation Management System based on BIM Technology (No. JC2021173).

**Data availability** The authors do not have permissions to share data.

## Declarations

**Conflict of interest** The authors declare no competing interests.

## References

Aïtcin P-C (1998) High performance concrete. CRC Press

Akopyan AV (2015) Geometry of the Cardioid. Am Math Mon 122:144–150

Alshihri MM, Azmy AM, El-Bisy MS (2009) Neural networks for predicting compressive strength of structural light weight concrete. Constr Build Mater 23:2214–2219

Benemaran RS, Esmaeili-Falak M (2020) Optimization of cost and mechanical properties of concrete with admixtures using MARS and PSO. Comput Concr 26:309–316. https://doi.org/10.12989/cac.2020.26.4.309

Bharatkumar B, Narayanan R, Raghuprasad B, Ramachandramurthy D (2001) Mix proportioning of high performance concrete. Cem Concr Compos 23:71–80. https://doi.org/10.1016/S0958-9465(00)00071-8

Cheng M-Y, Chou J-S, Roy AFV, Wu Y-W (2012) High-performance concrete compressive strength prediction using time-weighted evolutionary fuzzy support vector machines inference model. Autom Constr 28:106–115

Cheng H, Kitchen S, Daniels G (2022) Novel hybrid radial based neural network model on predicting the compressive strength of long-term HPC concrete. Adv Eng Intell Syst. https://doi.org/10.22034/aeis.2022.340732.1012

De Larrard F, Malier Y (2018) Engineering properties of very high performance concretes. High Perform Concr. CRC Press, pp 85–114

Hashim FA, Houssein EH, Hussain K, Mabrouk MS, Al-Atabany W (2022) Honey Badger algorithm: new metaheuristic algorithm for solving optimization problems. Math Comput Simul 192:84–110

Huang L, Jiang W, Wang Y, Zhu Y, Afzal M (2022) Prediction of long-term compressive strength of concrete with admixtures using hybrid swarm-based algorithms. Smart Struct Syst 29:433–444

Kapner DJ, Cook TS, Adelberger EG, Gundlach JH, Heckel BR, Hoyle CD et al (2007) Tests of the gravitational inverse-square law below the dark-energy length scale. Phys Rev Lett 98:21101

Khiabani MY, Sedaghat B, Ghorbanzadeh P, Porroustami N, Shahdany SMH, Yousef Hassani SN (2023) Application of a hybrid hydro-economic model to allocate water over the micro- and macro-scale region for enhancing socioeconomic criteria under the water shortage period. Water Econ Policy

Kisi O (2015) Pan evaporation modeling using least square support vector machine, multivariate adaptive regression splines and M5 model tree. J Hydrol 528:312–320

Kovačević M, Prištini S, Mitrovica K, Beogradu S, Ivanišević N, Dašić T et al (2016) Primjena neuronskih mreža za hidrološko modeliranje u krškom području 1:1–10

Kovačević M, Lozančić S, Nyarko EK, Hadzima-Nyarko M (2021) Modeling of compressive strength of self-compacting rubberized concrete using machine learning. Materials (bAsel) 14:4346

Lam L, Wong Y, Poon C (1998) Effect of fly ash and silica fume on compressive and fracture behaviors of concrete. Cem Concr Res 28:271–283. https://doi.org/10.1016/S0008-8846(97)00269-X

Lim C-H, Yoon Y-S, Kim J-H (2004) Genetic algorithm in mix proportioning of high-performance concrete. Cem Concr Res 34:409–420

Lyngdoh GA, Zaki M, Krishnan NMA, Das S (2022) Prediction of concrete strengths enabled by missing data imputation and interpretable machine learning. Cem Concr Compos 128:104414

Masoumi F, Najjar-Ghabel S, Safarzadeh A, Sadaghat B (2020) Automatic calibration of the groundwater simulation model with high parameter dimensionality using sequential uncertainty fitting approach. Water Supply 20:3487–3501. https://doi.org/10.2166/ws.2020.241

Naruei I, Keynia F (2021) A new optimization method based on COOT bird natural life model. Expert Syst Appl 183:115352

Nguyen H, Vu T, Vo TP, Thai H-T (2021) Efficient machine learning models for prediction of concrete strengths. Constr Build Mater 266:120950

Ni H-G, Wang J-Z (2000) Prediction of compressive strength of concrete by neural networks. Cem Concr Res 30:1245–1250

Paillisson J-M, Marion L (2001) Interaction between coot (*Fulica atra*) and waterlily (*Nymphaea alba*) in a lake: the indirect impact of foraging. Aquat Bot 71:209–216

Prasad BKR, Eskandari H, Reddy BVV (2009) Prediction of compressive strength of SCC and HPC with high volume fly ash using ANN. Constr Build Mater 23:117–128

Russell HG (1999) ACI defines high-performance concrete. Concr Int 21:56–57

Sarkhani-Benemaran R, Esmaeili-Falak M, Javadi A (2022) Predicting resilient modulus of flexible pavement foundation using extreme gradient boosting based optimised models. Int J Pavement Eng. https://doi.org/10.1080/10298436.2022.2095385

Sobhani J, Najimi M, Pourkhorshidi AR, Parhizkar T (2010) Prediction of the compressive strength of no-slump concrete: a comparative study of regression, neural network and ANFIS models. Constr Build Mater 24:709–718

Suykens JAK, De Brabanter J, Lukas L, Vandewalle J (2002) Weighted least squares support vector machines: robustness and sparse approximation. Neurocomputing 48:85–105

Topcu IB, Sarıdemir M (2008) Prediction of compressive strength of concrete containing fly ash using artificial neural networks and fuzzy logic. Comput Mater Sci 41:305–311

Yeh I-C (1998) Modeling concrete strength with augment-neuron networks. J Mater Civ Eng 10:263–268. https://doi.org/10.1061/(ASCE)0899-1561(1998)10:4(263)

Yin H, Liu S, Lu S, Nie W, Jia B (2021) Prediction of the compressive and tensile strength of HPC concrete with fly ash and micro-silica using hybrid algorithms. Adv Concr Constr 12:339–354

Zhang J, Li D, Wang Y (2020a) Toward intelligent construction: prediction of mechanical properties of manufactured-sand concrete using tree-based models. J Clean Prod 258:120665

Zhang Y, Jin Z, Mirjalili S (2020b) Generalized normal distribution optimization and its applications in parameter extraction of photovoltaic models. Energy Convers Manag 224:113301