Journal of Engineering and Applied Science

RESEARCH

Open Access

Strength properties prediction of RCA concrete via hybrid regression framework

Linlin Yu[1*]

*Correspondence:
17764190409@163.com

[1] Department of Building and Materials Engineering, Hubei University of Education, Wuhan 430000, Hubei, China

Abstract

High-performance concrete (HPC) is commonly utilized in the construction industry because of its strength and durability. The mechanical properties of HPC, specifically its compressive and tensile strength, are crucial indicators. Accurate prediction of concrete strength is crucial for optimizing the design as well as the performance of concrete structures. In this investigation, a novel approach for strength prediction of HPC is proposed, employing the Support Vector Regression (SVR) algorithm in conjunction with three optimizers: the Slime Mold Algorithm (SMA), Adaptive Opposition Slime Mold Algorithm (AOSM), and Equilibrium Slime Mold Algorithm (ESMA). The SVR algorithm is a robust machine-learning technique that has displayed promising results in various prediction tasks. The utilization of SVR allows for the effective modeling and prediction of the complex relationship between the strength properties of HPC and the influencing factors. To achieve this, a dataset comprising 344 samples of high-performance concrete was collected and utilized to train and assess the SVR algorithm. However, the choice of suitable optimization algorithms becomes crucial to enhance prediction accuracy and convergence speed. Through extensive experimentation and comparative analysis, the proposed framework's performance is evaluated using real-world HPC strength data. The results demonstrate that combining SVR with AOSM, ESMA, and SMA outperforms traditional prediction accuracy and convergence speed optimization methods. The suggested framework provides an effective and reliable solution for accurately predicting the compressive strength (CS) of HPC, enabling engineers and researchers to optimize the design and construction processes of HPC structures.

Keywords: High-performance concrete, Support vector regression, Slime mold algorithm, Adaptive opposition slime mold algorithm, Equilibrium slime mold algorithm

Introduction

Modern engineering structures predominantly utilize concrete as the most prevalent construction material [1]. In complex environments, the construction of concrete structures necessitates the use of HPC [2], which must meet elevated standards for workability, strength, and durability [3]. Possessing excellent durability, workability, and properties of strength, high-performance concrete (HPC) is a homogeneous composition of high-quality cement, water, aggregates, and active fine admixtures [4, 5].

Reducing the size and weight of concrete structures, minimizing material requirements, enhancing durability, and extending the service life of structures are among the advantages gained through the utilization of high-performance concrete (HPC) in various projects, such as bridge components and dams. The preparation of HPC involves the addition of mineral admixtures, chemical admixtures, and fibrous materials to the concrete mixture [6–8].

HPC has gained significant attention in the construction industry because of its exceptional strength, durability, and enhanced properties compared to conventional concrete. The strength of accurate prediction of HPC is of paramount importance for optimizing the design, construction, and maintenance of concrete structures [9–11]. Traditional empirical methods often fail to capture the complex relationships among the factors influencing concrete strength. Therefore, integrating advanced prediction models and optimization algorithms becomes crucial to achieve more accurate and reliable results [12–14].

The utilization of machine learning (ML) as well as artificial intelligence (AI) techniques has been extensively explored in the area of experimental mechanics, encompassing the study of structures and materials for a wide range of purposes [15–18]. Researchers have explored integrating ML algorithms to predict and generate anticipated results based on experimental data [9]. ML encompasses various learning methods, including unsupervised, supervised, semi-supervised, and reinforcement learning. In the case of HPC, ML techniques have been employed to address real-world HPC challenges effectively. Notable methods consist of support vector machine (SVM), artificial neural network (ANN), gene expression programming (GEP), multilayer perceptron neural network (MLP), and the multigroup approach for data management to predict the desired output data. ML has indicated great potential in predicting concrete strength in recent years [19–21]. Among these techniques, Support Vector Regression (SVR) [22] has emerged as a robust and efficient approach for modeling and forecasting complex nonlinear relationships. SVR has been successfully used in various fields because of its ability to handle high-dimensional data, handle nonlinearity, and mitigate the risk of overfitting [23]. While SVR has demonstrated promising results, optimizing its parameters is critical to further enhancing its predictive performance [24].

To this end, researchers have explored the integration of various optimization algorithms to enhance the accuracy as well as convergence speed of the SVR model. In this context, the Adaptive Opposition Slime Mold Algorithm (AOSMA), Equilibrium Slime Mold Algorithm (ESMA), and Slime Mold Algorithm (SMA) have emerged as effective optimizers in different domains. For appraising the performance of the suggested framework, a dataset containing 344 samples of high-performance concrete was collected. This dataset was used to train and evaluate the SVR algorithm integrated with the mentioned optimizers. Comparative analyses assessed the proposed approach's predictive accuracy and convergence speed against traditional optimization methods [25].

By utilizing statistical metrics like Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Relative Absolute Error (MRAE), Coefficient Correlation (R^2), and weight absolute percentage error (WAPE), the significance of optimization algorithms in enhancing the accuracy of the SVR prediction model is underscored. The SVR model's performance is evaluated both with and without these algorithms, and the outcomes demonstrate that the inclusion of optimization algorithms leads to superior performance compared to the model without them. Overall, this study aims to contribute to the advancement of accurate strength prediction models for high-performance concrete. Accurate strength prediction models for HPC are of immense importance in the field of civil engineering, as they allow for the optimization of various parameters, consisting of the selection of concrete mix ingredients and proportions, to achieve desired performance outcomes. By accurately predicting the strength properties of HPC, engineers can make informed decisions regarding structural design, material selection, and construction techniques, resulting in improved total performance, sustainability, and durability of concrete structures. Through advancing these prediction models, this study intends to empower engineers and researchers with robust tools that enable them to optimize the use of HPC, leading to structures that exhibit enhanced strength, resilience, and longevity. By optimizing the design and construction processes of concrete structures, the study ultimately contributes to improving the built environment, fostering sustainability, and positively impacting the infrastructure sector.

Methods

Data gathering

The data-gathering process for this study involved collecting information on various input variables that influence the strength properties of HPC. These parameters include Natural Coarse Aggregate (NCA), Water (W), Cement (C), Recycled Coarse Aggregate (RCA), Self-Compacting Recycled Aggregate (SRCA), Fine Aggregate (FA), Superplasticizer (SP), Chemical Admixtures (CS), Waste Recycled Concrete Aggregate (WRCA), and Dense Recycled Concrete Aggregate (DRCA) [26]. To establish a comprehensive dataset, a meticulous approach was employed to procure data from diverse sources. Extensive scrutiny of pertinent literature, research articles, industry standards, and concrete mix design databases was conducted to acquire information regarding the input parameters. In addition, collaboration with experts and practitioners in the field of concrete technology provided valuable insights and data sources. The collected data included various samples representing different HPC mix designs, geographical locations, and experimental conditions. Care was taken to include laboratory-tested and field data from real-world construction projects. The dataset encompassed sufficient samples to ensure statistical significance and cover a diverse range of HPC compositions. The compiled dataset is a valuable resource for training and evaluating HPC strength prediction models using the SVR and optimization algorithms. Table 1 shows the statistical properties of the dataset.

Support vector regression (SVR)

In the beginning, Vapnik VN developed the support vector machine (SVM) to address classification problems [22]. However, it was later improved to handle regression problems as well. Compared to the conventional Empirical Risk Minimization (ERM) principle, the structural risk minimization (SRM) principle is considerably more sophisticated and is followed by SVM regression [27]. Utilizing the SRM principle in statistical learning is crucial as it strives to minimize the upper bound generalization error, a

Table 1 Model input and target values statistical properties

Indicators	Variables									
	Input targets									
	Water (kg/m^3)	Cement (kg/m^3)	FA (kg/m^3)	NCA (kg/m^3)	RCA (kg/m^3)	SP (kg/m^3)	SRCA (mm)	DRCA (kg/m^3)	WRCA (%)	CS (MPa)
Max	271	600	1010	1448.25	1574.3	7.8	32	2661	10.9	108.5
Min	117.6	158	0	0	0	0	10	0	0	13.4
Average	184.62	386.86	681.88	398.07	596.35	1.3241	19.755	2231.0	4.8046	44.394
St. Dev	25.835	82.160	205.28	370.70	371.69	2.0512	4.0201	580.95	2.2624	15.617

pivotal aspect of the learning process. SVR, an extension of SVM, is applied to address regression issues [28]. Although SVR and SVM employ comparable algorithms, they are tailored to estimate different parameters. The primary discrepancy between the two methods is in implementing slack variables [29].

Linear support vector regression

Given a training dataset of $\{y_{i,} x_i, i=1, 2, 3 \ldots n\}$, where y_i represents the output vector, x_i represents the input vector, also n represents the dataset size, the local linear regression form of SVR can be represented as:

$$f(x,k) = k \times x + b \tag{1}$$

The Eq. above represents the dot product as (x, k), where k shows the vector of weight, x represents the normalized test pattern, and b shows the bias. To implement the SRM theory, the empirical risk R_{emp} (k, b) is minimized, which can be expressed by an equation. Equation (3) shows that the empirical risk is computed using an ε-insensitive loss function denoted by $L_\varepsilon(y_i, f(x_i, k))$.

$$R_{emp}(k,b) = \frac{1}{n}\sum_{i=1}^{n} L_\varepsilon(y_i, f(x_i,k)) \tag{2}$$

$$L_\varepsilon\big(y_i, f(x_i,k)\big) = \begin{cases} \varepsilon, if \left|y_i - f(x_i,k)\right| \le \varepsilon \\ \left|y_i - f(x_i,k)\right| - \varepsilon, otherwise \end{cases} \tag{3}$$

During the optimization process, the ε-insensitive loss function, denoted as $L_\varepsilon\big(y_i, f(x_i,k)\big)$, measures the tolerance error between the target output y_i and the estimated output values $f(x_i, k)$. The training pattern, x_i, is also defined in this context. In linear regression problems using the ε-insensitive loss function, minimizing the squared norm of the weight vector, $\|k\|^2$, can simplify the complexity of the SVR model. Additionally, a non-negative slack variable $\left(\varphi_i^* \varphi_i\right)$ can be utilized to estimate the divergence of the training data outside the ε-insensitive zone, represented by φ_i.

$$\begin{aligned} &\underset{k,b,\varphi,\varphi^*}{\text{Lim}} \left[\tfrac{1}{2}k.k + c\left(\sum_{i=1}^{n}\varphi_i^* + \sum_{1=1}^{n}\varphi_i\right)\right] \\ &\text{Subjected to,} \begin{cases} y_i - k.x_i - b \le \varepsilon + \varphi_i^* \\ k.x_i + b - y_i \le \varepsilon + \varphi_i \;\; i = 1, \ldots, n \\ \varphi_i^*, \varphi_i \ge 0 \end{cases} \end{aligned} \tag{4}$$

To solve the previously mentioned problem, locating the Lagrange function saddle point is necessary.

$$\begin{aligned} L(k,\varphi^*,\varphi,\alpha^*,\alpha,c,\gamma^*,\gamma) = {}& \frac{1}{2}k.k + c\left(\sum_{i=1}^{n}\varphi_i^* + \sum_{1=1}^{n}\varphi_i\right) - \sum_{i=1}^{n}\alpha_i[y_i - k.x_i - b + \varepsilon + \varphi_i] \\ & - \sum_{i=1}^{n}\alpha_i^*\left[k.x_i + b - y_i + \varepsilon + \varphi_i^*\right] - \sum_{1}^{n}(\gamma_i^*\varphi_i^* + \gamma_i\varphi_i) \end{aligned} \tag{5}$$

The Lagrange function can be minimized through the application of the KKT conditions, which involves performing partial differentiation of Eq. (5) concerning k, b, φ_i^*, and φ_i.

$$\frac{\delta \mathrm{L}}{\delta \mathrm{k}} = \mathrm{k} + \sum_{i=1}^{n} \alpha_i x_i - \sum_{i=1}^{n} \alpha_i^* x_i = 0, k = \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) x_i \tag{6}$$

$$\frac{\delta \mathrm{L}}{\delta \mathrm{b}} = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i^* = 0, \sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \alpha_i^* \tag{7}$$

$$\frac{\delta \mathrm{L}}{\delta \varphi^*} = \mathrm{c} - \sum_{i=1}^{n} \gamma_i^* - \sum_{i=1}^{n} \alpha_i^* = 0, \sum_{i=1}^{n} \gamma_i^* = c - \sum_{i=1}^{n} \alpha_i^* \tag{8}$$

$$\frac{\delta \mathrm{L}}{\delta \varphi} = \mathrm{c} - \sum_{i=1}^{n} \gamma_i - \sum_{i=1}^{n} \alpha_i = 0, \sum_{i=1}^{n} \gamma_i = c - \sum_{i=1}^{n} \alpha_i \tag{9}$$

Here, k is linked to the parameter k in Eq. (1). Substituting Eq. (6) into the Lagrange function (5) yields the dual optimization function, presented in:

$$\begin{aligned} & max_{\alpha,\alpha^*}[k(\alpha,\alpha^*)] = max_{\alpha,\alpha^*}\left[\sum_{i=1}^{n} \gamma_i(\alpha_i^* - \alpha_i) - \varepsilon \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) - \frac{1}{2} \sum_{ij=1}^{n} (\alpha_i^* - \alpha_i)(\alpha_i^* - \alpha_i)(x_i.x_j)\right] \\ & subjected\ to \begin{cases} \sum_{i=1}^{n} (\alpha_i^* - \alpha_i) = 0 \\ 0 \le \alpha_i^*, \alpha_i \le 0 \end{cases} \quad i = 1, \ldots, n \end{aligned} \tag{10}$$

The Lagrange multiplier α_i^* and α_i are used to define the optimization problem [30]. Once Eq. (10) is solved under the constraints in Eq. (11), the ultimate linear regression function can be stated as:

$$f(x, \alpha^*, \alpha) = \sum_{i=1}^{n} (\alpha_i^* - \alpha_i)(x_i, x) + b \tag{11}$$

Nonlinear support vector regression

Linear SVR may not be suitable for complicated real-world problems. To address this, nonlinear SVR can be implemented by mapping the input data into a feature space with a high dimensional data, where linear regression can be utilized. To transform the input training algorithm, x_i, into the feature space, $\tau(x_i)$, the function of nonlinearity is utilized. The algorithm is then utilized in a similar way to linear SVR. As a result, the formulation of nonlinear SVR is represented as shown below:

$$f(x, k) = k \times \tau(x) + b \tag{12}$$

The parameter vector is denoted by k and b, while the mapping function $\tau(x)$ is employed to convert input features into a feature space with higher dimensionality.

The diagram in Fig. 1 depicts the nonlinear SVR with an ε-insensitive loss function. The bold points represent the support vectors, which have the maximum distance from the decision boundary.

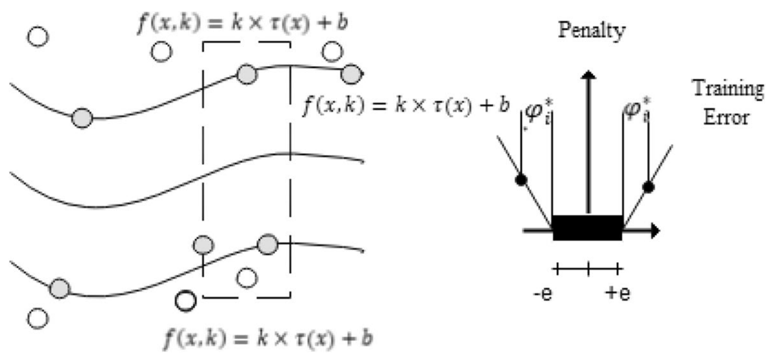
$f(x,k) = k \times \tau(x) + b$
$f(x,k) = k \times \tau(x) + b$
$f(x,k) = k \times \tau(x) + b$
Penalty
φ_i^*
φ_i^*
Training Error
-e
+e

Fig. 1 The insensitive loss function of nonlinear SVR

The ε-insensitive loss function, which is depicted on the right side of Fig. 1, includes an error tolerance ε and upper and lower bounds computed using the slack variable (φ_i^*, φ_i). In summary, nonlinear SVR may be represented in the following manner:

$$max_{\alpha,\alpha^*}[k(\alpha,\alpha^*)] = max_{\alpha,\alpha^*}\left[\sum_{i=1}^{n}\gamma_i\left(\alpha_i^*-\alpha_i\right) - \varepsilon\sum_{i=1}^{n}\left(\alpha_i^*+\alpha_i\right) - \frac{1}{2}\sum_{ij=1}^{n}\left(\alpha_i^*-\alpha_i\right)\left(\alpha_i^*-\alpha_i\right)\left(\tau(x_i).\tau\left(x_j\right)\right)\right]$$
$$subjected\ to\begin{cases}\sum_{i=1}^{n}\left(\alpha_i^*-\alpha_i\right)=0 & i=1,\ldots,n\\ 0\le\alpha_i^*,\alpha_i\le 0\end{cases} \tag{13}$$

Due to the complexity of the inner product $\tau(x_i).\tau\left(x_j\right)$, it is possible to substitute it with the kernel function $\tau(x_i).\tau\left(x_j\right) = H\left(x_i.x_j\right)$.

$$max_{\alpha,\alpha^*}[k(\alpha,\alpha^*)] = max_{\alpha,\alpha^*}\left[\sum_{i=1}^{n}\gamma_i\left(\alpha_i^*-\alpha_i\right) - \varepsilon\sum_{i=1}^{n}\left(\alpha_i^*+\alpha_i\right) - \frac{1}{2}\sum_{ij=1}^{n}\left(\alpha_i^*-\alpha_i\right)\left(\alpha_i^*-\alpha_i\right)H(x_i.x_j)\right]$$
$$subjected\ to\begin{cases}\sum_{i=1}^{n}\left(\alpha_i^*-\alpha_i\right)=0 & i=1,\ldots,n\\ 0\le\alpha_i^*,\alpha_i\le 0\end{cases} \tag{14}$$

Slime Mold Algorithm (SMA)

This optimizer focuses on Physarum polycephalum, a type of slime mold. The mold's main nutrition phase is the Plasmodium stage, representing the dynamic and active stage. The slime mold's organic material aggressively searches for food during this stage [31], envelops it, and then secretes several enzymes to aid in its breakdown and digestion. Because of their inherent characteristics and unique patterns, these organisms can construct a venous network connecting multiple food sources simultaneously. By utilizing both negative and positive feedback mechanisms, the slime organism can efficiently determine the best possible route for connecting to food [32]. As a result, in the study of graph theory and path networks, the use of mathematical modeling and slime mold's practical implementation has been examined. This section will provide a detailed description of the proposed mathematical model and method [33].

To mimic the contraction mode of slime mold's approach behavior, the following formulas are proposed, as it can locate food according to the odor in the air:

$$\overrightarrow{X(t+1)} = \begin{cases} \overrightarrow{X_b(t)} + \overrightarrow{vb}.\left(\overrightarrow{U}.\overrightarrow{X_C(t)} - \overrightarrow{X_D(t)}\right), r < p \\ \overrightarrow{vc}.\overrightarrow{X(t)}, r \geq p \end{cases} \tag{15}$$

The parameter X denotes the position of the slime mold, $\overrightarrow{X_b}$ denotes the location of the individual with the highest odor concentration currently detected , $\overrightarrow{vb}$ has a range of $-a$ to a, and $\overrightarrow{vc}$ decreases linearly from one to zero. Here, t denotes the present iteration, $\overrightarrow{U}$ represents the weight of the slime mold, while the variables $\overrightarrow{X_C}$ and $\overrightarrow{X_D}$ denote two randomly chosen individuals from the slime mold. The formula for p is given below:

$$p = \tanh|S(i) - DF| \tag{16}$$

The DF illustrates the best fitness gained across all iterations, and variables $i \in 1, 2, \ldots, n$ while (i) illustrate the fitness of X. The formula for $\overrightarrow{vb}$ is given below:

$$\overrightarrow{vb} = [-a, a] \tag{17}$$

$$a = \operatorname{arctanh}(-(\frac{t}{\max\_t}) + 1) \tag{18}$$

The formula for $\overrightarrow{U}$ is presented below:

$$\overrightarrow{U(SmellIndex(i))} = \begin{cases} 1 + r.\log\left(\frac{bF - S(i)}{bF - wF} + 1\right), condition \\ 1 - r.\log\left(\frac{bF - S(i)}{bF - wF} + 1\right), other \end{cases} \tag{19}$$

$$SmellIndex = sort(S) \tag{20}$$

$\overrightarrow{U}$ is defined by the following formula: a random value in the range of 0 to 1 is denoted by r, bF denotes the optimum fitness achieved in the present iteration, wF illustrates the worst fitness value acquired in the current iterative procedure, and *SmellIndex* is the fitness values pattern arranged in ascending order for minimizing the value. Additionally, *condition* denotes that (i) ranks in the first half of the population.

Equation (21) is a mathematical model that simulates how the venous tissue anatomy of slime mold contracts during its search for food. The model is based on the interplay between vein width and food concentration, where a thicker vein corresponds to a stronger wave initiated by the faster cytoplasm and flow bio-oscillator. Equation (19) introduces the variable r to account for the venous contraction mode uncertainty. The use of the component *log* serves to stabilize the numerical values of the contraction frequency. The variable *condition* models the slime mold's ability to adapt its search pattern based on the food's quality. Specifically, when the food concentration is high, the weight of the nearby region increases; when it is low, the region's weight decreases, prompting the exploration of alternative areas.

Drawing upon the aforementioned principle, updating the location of slime mold, a mathematical formula can be expressed as follows:

$$\overrightarrow{X^*} = \begin{cases} rand.(UB - LB) + LB, rand < z \\ \overrightarrow{X_b(t)} + \overrightarrow{vb}.\left(U.\overrightarrow{X_C(t)} - \overrightarrow{X_D(t)}\right), r < p \\ \overrightarrow{vc}.\overrightarrow{X(t)}, r \geq p \end{cases} \tag{21}$$

The lower and upper boundaries of the search range are represented by *LB* and *UB*, respectively. *rand* and *r* denote random values within the interval of [0,1]. Algorithm 1 displays the pseudo-code for the SMA.

```
Initialize the parameters pop size, $Max\_iteraition$;
Initialize the positions of the slime mold $X_i (i = 1,2,\ldots,n)$;
**While** ($t \leq Max\_iteraition$)
    Compute the fitness of all slime mold;
    $update\ bestFitness, X_b$
    Compute the U by Eq. (19);
    **For** $each\ search\ portion$
        $update\ p, vb, vc$;
        $update\ positions$ by Eq. (22);
    **End For**
    $t = t + 1$;
**End While**
**Return** $bestFitness, X_b$;
```

Algorithm 1. SMA Algorithms pseudo-code

Equilibrium Slime Mold Algorithm (ESMA)

The foraging behavior of slime mold presents a promising origin of innovation for developing effective and efficient optimization methods [34]. The starting position vector of each slime mold is randomly generated through a randomization process.

$$\overrightarrow{\mathrm{X}}_i(t = 1) = r_1.(UB - LB) + LB,\ i = 1, 2, \ldots, N. \tag{22}$$

The positioning model for the ith slime mold is represented as X_i $(j = 1, 2, ..., N)$, in the next iteration $(t + 1)$, is established utilizing SMA as follows:

$$\overrightarrow{\mathrm{X}}_i(t=1) = \begin{cases} r_1.(UB - LB) + LB, r_1 < z \\ \overrightarrow{\mathrm{X}}_{Gbest} + \overrightarrow{step_a}.\left(\overrightarrow{\mathrm{U}}.\overrightarrow{\mathrm{X}}_C - \overrightarrow{\mathrm{X}}_D\right), r_2 < P_i(t) and r_1 \geq z \\ \overrightarrow{step_b}.\overrightarrow{\mathrm{X}}_i(t), r_2 \geq P_i(t) and r_1 \geq z \end{cases} \tag{23}$$

The $\overrightarrow{\mathrm{X}}_{Gbest}$ denotes the value of the global best fitness achieved across iterations one to t. Additionally, the variables r_1 and r_2 correspond to random values within the range of [0, 1].

To eradicate and disseminate the slime mold, a probability denoted by z is utilized. Within the context of this study, z is a constant value of 0.03 [35]. Equation (24) is utilized to sort the fitness values in ascending order.

$$[sortf, sortIndex] = sort\left(f\right), where\, f = \{f_1, f_2, \ldots., f_N\} \tag{24}$$

Equation (25) is employed to calculate $\overrightarrow{\mathrm{U}}$.

$$\overrightarrow{\mathrm{U}}(sortIndex(j)) = \begin{cases} 1 + r_3.\log\left(\frac{f_{Lbest} - sortf(j)}{f_{Lbest} - f_{Lworst}} + 1\right) 1 \leq j \leq \frac{N}{2} \\ 1 - r_3.\log\left(\frac{f_{Lbest} - sortf(j)}{f_{Lbest} - f_{Lworst}} + 1\right) \frac{N}{2} < j \leq N \end{cases} \tag{25}$$

within the range of [0,1], a random number,r_3, uniformly distributed, is utilized. The local worst and best fitness values acquired during the present iteration are denoted by f_{Lworst} and f_{Lbest}, respectively. Equations (26–27) are employed to calculate these fitness values.

$$f_{Lbest} = sortf(1) \tag{26}$$

and

$$f_{Lworst} = sortf(N) \tag{27}$$

Below is the formula that defines the variable P_i, which represents the probability of choosing the ith slime mold's trajectory:

$$P_i = \tanh\left|f(X_i) - f_{Gbest}\right| \tag{28}$$

For each$i = 1, 2, ..., N$, the fitness value of the i-th slime mold in X_i is determined by $f(X_i)$. The first iteration's global best fitness value up to the present iteration is represented byf_{Gbest}. The magnitude of the step size is indicated by $\overrightarrow{step_a}$ and is determined by a uniform distribution ranging from − a to a. Similarly, the size of the step, represented by$\overrightarrow{step_b}$, is determined by a uniform distribution ranging from − b to b. The values of a and b are determined by Eq. (30), which is a function of the current iteration t as well as the maximum iteration T:

$$a = \operatorname{arctanh}(-\left(\frac{t}{T}\right) + 1) \tag{29}$$

and

$$b = 1 - \frac{t}{T} \tag{30}$$

Despite the SMA's promising results, there is still room for improvement in the search process, as indicated by Eq. (24). It is essential to note that incorporating random slime molds can alter the trajectory of the search. Local minima can constrain the efficacy of the search process when selecting individuals $\overrightarrow{X}_D$ and $\overrightarrow{X}_C$ from a sample of N slime molds. This section introduces a new optimization technique called the Equilibrium Slime Mold Algorithm (EOSM). This algorithm replaces the position vector $\overrightarrow{X}_A$ with a vector derived from an equilibrium pool of four superior position vectors. The Equilibrium Optimizer (EO) concept is then used to calculate the average position of this selection. Equation (31) precisely defines the components of the equilibrium pool.

$$\begin{aligned}
&\overrightarrow{X}_{eq(1)} = X(sortIndex(1))\\
&\overrightarrow{X}_{eq(2)} = X(sortIndex(2))\\
&\overrightarrow{X}_{eq(3)} = X(sortIndex(3))\\
&\overrightarrow{X}_{eq(4)} = X(sortIndex(4))\\
&\overrightarrow{X}_{ave} = \frac{\overrightarrow{X}_{eq(1)}+\overrightarrow{X}_{eq(2)}+\overrightarrow{X}_{eq(3)}+\overrightarrow{X}_{eq(4)}}{4}
\end{aligned} \tag{31}$$

A set of five-position vectors is utilized to construct the equilibrium pool, represented by $\overrightarrow{X}_{eq,pool}$.

$$\overrightarrow{X}_{eq,pool} = \left\{\overrightarrow{X}_{eq(1)}, \overrightarrow{X}_{eq(2)}, \overrightarrow{X}_{eq(3)}, \overrightarrow{X}_{eq(4)}, \overrightarrow{X}_{ave}\right\} \tag{32}$$

In ESMA, the position vector for the ith slime mold,$X_i(j = 1, 2, ..., N)$, during the fresh iteration $(t + 1)$ is represented by the following equations:

$$\overrightarrow{X}_i(t+1) = r_1.(UB - LB) + LB, \ when\ r_1 < z \tag{33a}$$

$$\overrightarrow{X}_i(t+1) = \overrightarrow{X}_{Gbest} + \overrightarrow{step_a}.\left(\overrightarrow{U}.\overrightarrow{X}_{eq} - \overrightarrow{X}_D\right), \ when\ r_2 < p_i(t)\ and\ r_1 \geq z \tag{33b}$$

$$\overrightarrow{X}_i(t+1) = \overrightarrow{step_b}.\overrightarrow{X}_i(t), \ when\ r_2 \geq p_i(t)\ and\ r_1 \geq z \tag{33c}$$

The position vector $\overrightarrow{X}_{eq}$ is obtained by randomly selecting a vector from the equilibrium pool. The algorithmic tool z is employed to facilitate exploration in the search process, ensuring ESMA's effectiveness by preventing minimal local occurrence. An experimentally determined threshold value of 0.03 is utilized to achieve this objective. It is important to note that the ESMA algorithm modifies the position vector in the following iteration through a combination of the global best position, the local best position obtained from the best-so-far equilibrium pool, as well as a random vector. This approach allows for a balanced exploration–exploitation trade-off. Algorithm 2 details the proposed ESMA.

```
Begin
    Inputs: UB, LB, D, N, T, and z.
    Initialization: $x_i = \{x_i^1, x_i^2, \ldots, x_i^D\}$ for $i = 1, 2, \ldots, N$ at a random position within the search boundary [LB, UB] for a D dimension at initial iteration t=1 to form $X = [\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_i, \ldots, \vec{X}_N]'$.
    While $(t \leq T)$
    → Analyze the fitness f(x) of N slime mold.
    → The fitness value is sorted using Eq. (31) into ascending order (for the minimization issue), the $f_{Lbest}$ and $f_{Lworst}$ are determined using Eq. (33) (34), and the $\vec{X}_{eq,pool}$ is built using Eq. (32).
    → Determine the weighting factor $\vec{U}$ utilizing Eq. (32).
    → Update the $\vec{X}_{Gbest}$.
    → Determine the $a$ and $b$.
    For $i = 1$ : quantity of slime mold (N)
        → Generate a random value $r_1$.
        If $r_1 < z$
            • Update the position vector $\vec{X}_i(t+1) = r_1.(UB - LB) + LB$.
        Else If $r_1 \geq z$
            • Update the probability $p_i$, $\overrightarrow{step_a}$ and $\overrightarrow{step_b}$.
            • Pick one position vector $\vec{X}_{eq}$ at random from the equilibrium pool.
            • Create a random value $r_2$.
            If $r_2 < p_i$
                • Choose a random position $\vec{X}_D$ a vector from X.
                • Update the position vector $\vec{X}_i(t+1) = \vec{X}_{Gbest} + \overrightarrow{step_a}.(\vec{U}.\vec{X}_{eq} - \vec{X}_D)$.
            Else If $r_2 \geq p_i$
                • Update the position vector $\vec{X}_i(t+1) = \overrightarrow{step_b}.\vec{X}_i(t)$.
            End If
        End If
    End For
    → $t = t + 1$
    End while
    Output: $\vec{X}_{Gbest}$
End
```

Algorithm 2. ESMA Algorithms pseudo-code

Adaptive Opposition Slime Mold Algorithm (AOSM)

The wavering mode of plasmodial slime mold (Physarum polycephalum) is utilized by the SMA, a stochastic optimizer, to find the optimal solution to a given function. On the other hand, the AOSM algorithm incorporates opposition-based learning with an adaptive decision-making process that modifies the slime mold's approach behavior to suit the environment better, resulting in better solutions for a wider range of problems. Both algorithms are examples of bio-inspired optimization algorithms that leverage the behavior of biological organisms to solve complicated optimization problems more effectively and efficiently. To connect with food, slime mold utilizes positive–negative feedback and the oscillation mode to determine the optimal path. The position of the *i*th slime mold in *d*-dimensions can be represented as $Xi = (x_i^1, x_i^2, \cdots, x_i^d)$, where *i* is an element of the range [1, *N*]. The fitness or odor of the *i*th slime is denoted $byf(Xi), for \forall i \in [1, N]$. Suppose there are *N* slime molds in the search space, with a lower boundary of (*LB*) as well as an upper boundary of (*UB*). Therefore, the fitness and location of *N* slime molds at the present iteration *t* can be represented as:

$$X(t) = \begin{bmatrix} x_1^1 & x_1^2 & \ldots & x_1^d \\ x_2^1 & x_2^2 & \ldots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^d & \cdots & x_n^d \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{34}$$

$$f(x) = [f(x_1), f(x_2), \ldots, f(x_3)] \tag{35}$$

Equation (36) is employed in SMA to update the location of the slime mold for the subsequent iteration $(t+1)$.

$$X_i(t+1) = \begin{cases} rand.(UB - LB) + LB, r_1 < z \\ X_{lb}(t) + vb.(W.X_A(t) - X_B(t)), r_1 \geq \delta \text{ } and \text{ } r_2 < p_i \\ vc.x_i(t), r_1 > \delta \text{ } and \text{ } r_2 \geq p_i \end{cases} \tag{36}$$

W is the weight factor, and *Vb* and *Vc* are the random velocity factors. X_{LB} represented the individual with the best fitness value among the local population. In the present population, X_A and X_B are chosen randomly to represent two different slime molds [36]. The probability of the slime mold initializing at a random search location is denoted by δ, which remains fixed at 0.03. Random numbers r_1 and r_2 are generated within the range of [0,1]. *pi* is the threshold value of the *i*th slime mold that determines whether to use the best individual or its own position for the subsequent iteration. It can be computed as follows:

$$P_i = \tanh\left|f(X_i) - f_{Gbest}\right|, \forall i \in [1, N] \tag{37}$$

The global optimal fitness value f_{Gbest}, which is based on the global best position X_{Gbest}, is obtained using Eq. (38). (*Xi*) denotes *i*th slime mold's fitness value, *Xi*.

$$f_{Gbest} = f(X_{Gbest}) \tag{38}$$

In the present iteration *t*, use the following Eq. to calculate the weight *W* for *N* slime molds, use the following Eq.:

$$\mathrm{W}(sortIndf(i)) = \begin{cases} 1 + rand \;.\; \log\left(\frac{f_{Lbest} - f(X_i)}{f_{Lbest} - f_{Lworst}} + 1\right) 1 \leq i \leq \frac{N}{2} \\ 1 - rand \;.\; \log\left(\frac{f_{Lbest} - f(X_i)}{f_{Lbest} - f_{Lworst}} + 1\right) \frac{N}{2} < i \leq N \end{cases} \tag{39}$$

When solving a minimization issue, the fitness values are arranged in ascending order, as displayed below. Then, the weight W is computed using the given Eq., where *rand* is a random number between 0 and 1, f_{Lworst} represents the local worst fitness value , andf_{Lbest} represents the local best fitness value. Both of these values are derived according to the fitness value.*f*, as defined in Eq. (40):

$$\left[sort_f, sortlnd_f\right] = sort(f) \tag{40}$$

To determine its corresponding local best individual X_{Lbest} and the best local fitness value f_{Lbest} Follow the steps below:

$$f_{Lbest} = f(sortf(1)) \tag{41}$$

$$X_{Lbest} = x(sortf(1)) \tag{42}$$

To obtain the local worst fitness value*fLW*, follow the steps below:

$$f_{Lworst} = f(sortf(N)) \tag{43}$$

The random velocity factors *Vb* and *Vc* are obtained from a continuous uniform distribution in the intervals of [− *b*, *b*] and [− *c*, *c*], respectively. To determine the values of *b* and *c* for the current iteration *t*, use the following procedure:

$$b = \operatorname{arctanh}(-\left(\frac{t}{T}\right) + 1) \tag{44}$$

and

$$c = 1 - \frac{t}{T} \tag{45}$$

To enhance convergence and prevent getting stuck in local minima, opposition-based learning (OBL) is utilized. In OBL, the position Xn_i of each slime mold (where$i = 1, 2, \cdots, N$) in the search space is compared with its exact opposite positionXo_i. The difference is estimated to update the position for the succeeding iteration. The estimated value of Xo_i for the *i*th slime mold in the *j*th dimension is calculated using the following formula:

$$\begin{gathered} Xo_i^j(t) = \min(Xn_i(t)) + \max(Xn_i(t)) - Xn_i^j(t) \\ where\ i = 1, 2, \ldots, Nandj = 1, 2, \ldots, d. \end{gathered} \tag{46}$$

Xsi can be defined as the position of the *i*th slime mold chosen for the minimization problem.

$$Xs_i(t) = \begin{cases} Xo_i(t)\ if\ f(Xo_i(t)) < f(Xn_i(t)) \\ Xn_i(t)\ if\ f(Xo_i(t)) \geq f(Xn_i(t)) \end{cases} \tag{47}$$

When the slime mold is following a nutrient path that has been explored before, an adaptive decision strategy is used. This strategy considers both the corresponding previous fitness value ((t)) and the current fitness value ($Xni(t)$). To enable additional exploration as needed, OBL is incorporated into the adaptive decision strategy of AOSM. The updated position for the next iteration is determined utilizing this strategy, which can be represented as follows:

$$X_i(t+1) = \begin{cases} Xn_i(t)\ if\ f(Xn_i(t)) \leq f(X_i(t)) \\ Xs_i(t)\ if\ f(Xn_i(t)) > f(X_i(t)) \end{cases} \tag{48}$$

By utilizing an adaptive decision strategy to assess the necessity of OBL during the search trajectory, the AOSM method effectively enhances the efficiency of SMA. In addition, Fig. 2 shows the flowchart of AOSM.

Performance evaluators

This section provides various measures to assess hybrid models by measuring their degree of error and correlation. The metrics covered here comprise Mean Relative Absolute Error (MRAE), Root Mean Square Error (RMSE), Coefficient Correlation (R^2), weight absolute percentage error (WAPE), and Mean Absolute Error (MAE). The formulas for each of these metrics are listed below.

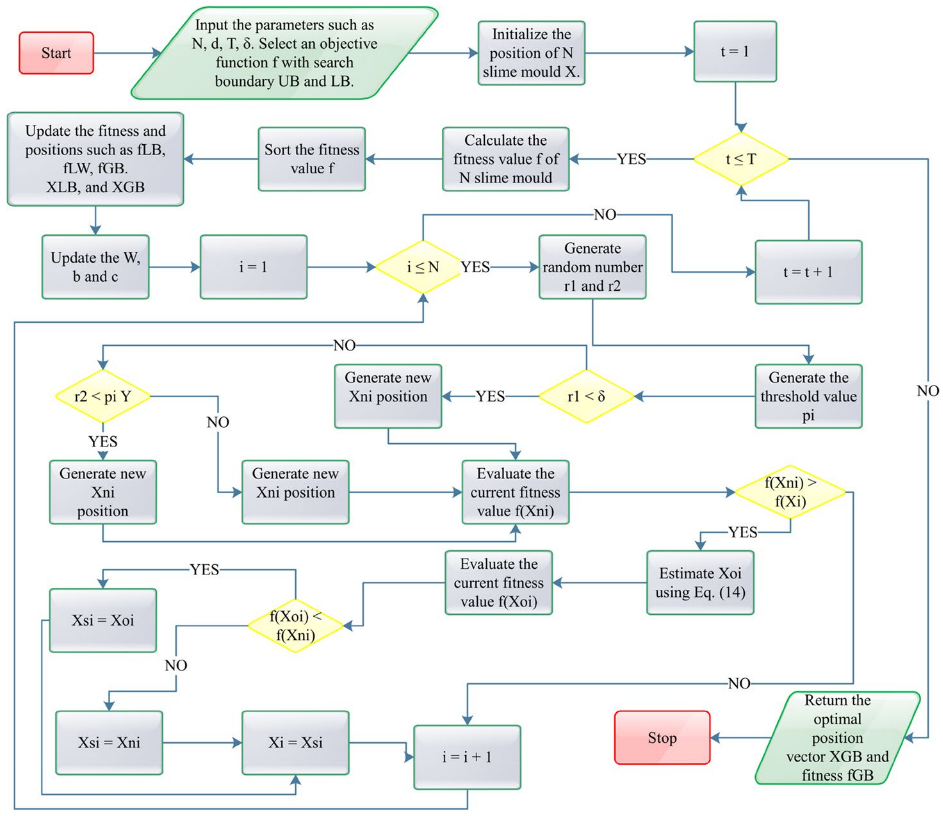
Start
Input the parameters such as N, d, T, δ. Select an objective function f with search boundary UB and LB.
Initialize the position of N slime mould X.
t = 1
t ≤ T
YES
Calculate the fitness value f of N slime mould
Sort the fitness value f
Update the fitness and positions such as fLB, fLW, fGB. XLB, and XGB
Update the W, b and c
i = 1
i ≤ N
YES
NO
Generate random number r1 and r2
t = t + 1
NO
Generate the threshold value pi
r1 < δ
YES
Generate new Xni position
NO
r2 < pi Y
YES
NO
Generate new Xni position
Generate new Xni position
Evaluate the current fitness value f(Xni)
f(Xni) > f(Xi)
YES
Estimate Xoi using Eq. (14)
Evaluate the current fitness value f(Xoi)
f(Xoi) < f(Xni)
YES
Xsi = Xoi
NO
Xsi = Xni
Xi = Xsi
i = i + 1
NO
Stop
Return the optimal position vector XGB and fitness fGB

Fig. 2 Flowchart of AOSM

$$R^2 = \left(\frac{\sum_{i=1}^{n} \left(b_i - \overline{b}\right)(m_i - \overline{m})}{\sqrt{\left[\sum_{i=1}^{n} \left(b_i - \overline{b}\right)^2\right] \left[\sum_{i=1}^{n} (m_i - \overline{m})^2\right]}} \right)^2 \tag{49}$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (m_i - b_i)^2} \tag{50}$$

$$MAE = \frac{1}{n} \sum_{i=1}^{n} \left| b_i - m_i \right| \tag{51}$$

$$WAPE = \max \left[\frac{\left| b_i - m_i \right|}{b_i} \right] \tag{52}$$

$$MRAE = \frac{1}{n} \sum_{i=1}^{n} \frac{\left| e_j \right|}{\left| A_j - \overline{A} \right|} \tag{53}$$

Equations (49–53) use the variable n to indicate the samples's number, b_i to represent the predicted value, $\overline{b}$ and $\overline{m}$ to denote the mean measured and predicted values, respectively, and m_i to indicate the measured value alternatively.

Results and discussion

In this research, three models were used to predict CS, and their performance was evaluated against experimental measurements obtained during the testing and training phases. The models employed were SVR-adaptive opposition slime mold algorithm (SVAM), SVR-equilibrium slime mold algorithm (SVES), and SVR-slime mold algorithm (SVSM). Five statistical metrics (R^2, RMSE, WAPE, MRAE, and MAE) were employed to assess and contrast the algorithms used in this investigation comprehensively. The experimental data was split into testing (30%) and training (70%) sets, as shown in Table 2, to ensure that the models were assessed on previously unseen data and provide an unbiased evaluation of their performance. A high R^2 value close to 1 indicates the excellent performance of the algorithm in both the testing and training phases, while

Table 2 Developed assessment outcomes of models by evaluators

Models	SVAM		SVES		SVSM	
States	Train	Test	Train	Test	Train	Test
R^2	*0.9779*	*0.9855*	*0.9908*	*0.9894*	*0.9740*	*0.9687*
RMSE	*2.3236*	*2.2261*	*1.4815*	*1.8396*	*2.7076*	*3.5590*
MAE	*1.2973*	*1.2551*	*0.646*	*0.8506*	*1.1112*	*1.364*
WAPE	*0.0293*	*0.028*	*0.0146*	*0.019*	*0.0251*	*0.0304*
MRAE	*0.3623*	*0.2499*	*0.1495*	*0.1752*	*0.24*	*0.1976*

lower values of metrics such as RMSE, WAPE, MRAE, and MAE indicate a desirable error level in the model. These metrics were used to evaluate the effectiveness of the algorithms used in this study.

The three hybrid models, SVAM, SVES, and SVSM, were evaluated based on their performance in predicting the High-Performance Concrete (HPC) properties. The models in question were assessed using the R^2 value, a statistical measure indicating the amount of variance in the dependent variable that the independent variable can explain. SVES exhibited the highest R^2 values, with 0.9908 and 0.9894 in the training and testing phases, respectively, indicating outstanding predictive accuracy. SVAM also performed well, with R^2 values of 0.9779 and 0.9855 in the training and testing phases, respectively, demonstrating high predictive accuracy. Meanwhile, SVSM showed slightly lower R^2 values of 0.970 and 0.9687 in the train and test phases but still demonstrated acceptable predictive accuracy.

While R^2 is a valuable metric for evaluating model performance, it should not be the sole criterion for assessment. Critical metrics such as RMSE, WAPE, MRAE, and MAE should also be considered to manage a more comprehensive evaluation. The outcomes show that the SVES model demonstrated lower error indicators in both the testing and training phases, suggesting better performance compared to the SVSM and SVAM models. Conversely, the SVSM model generally performed poorly, with higher error indicators and the lowest R^2 values. Overall, the outcomes suggest that SVES and SVAM may be better suited for CS prediction of HPC than SVSM. However, it is worth noting that other factors, such as model complexity, computational efficiency, and ease of implementation, should also be considered when selecting a model for practical applications. Nonetheless, the outcomes of this comparison provide valuable insights into the relative strengths and weaknesses of these three hybrid models for predicting HPC properties.

In Fig. 3, a scatter plot is presented, which compares the predicted values with the actual values for three hybrid models: SVAM, SVES, and SVSM. The scatter plot contains a center line and two linear fits representing the testing and training phases. It can be observed that all three models show a strong positive correlation between the actual and predicted values. However, SVES displays the most tightly clustered data points around the linear fit lines, indicating that it is the most accurate of the three models. Although SVAM and SVSM exhibit a strong correlation, their data points are slightly more scattered. The linear fit lines for both models have a similar slope and intercept,

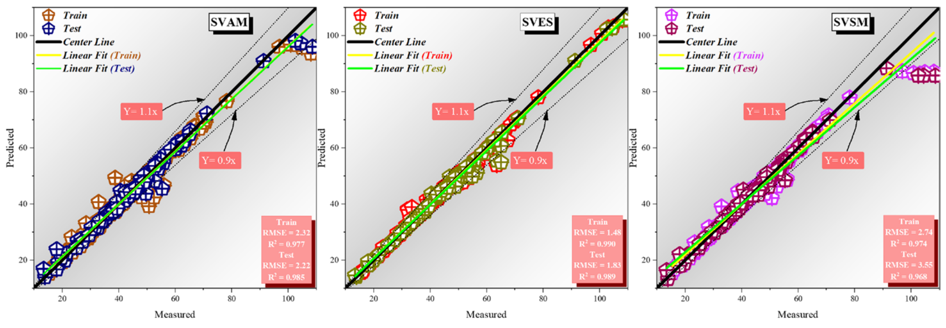

Fig. 3 The scatter plot for the hybrid models in the testing and training phase

indicating that their predictive capabilities are similar. Overall, the scatter plot visually represents the models' performance and highlights that SVES is the most effective in predicting CS.

Figure 4 presents a line series plot that compares the performance of three models (SVAM, SVES, and SVSM) in predicting high-performance concrete (HPC) strength. The *x*-axis represents the measured and training data, while the *y*-axis represents the compressive strength of concrete (CS) in mega-pascals (MP). The plot shows that all three models can accurately predict HPC strength, with the predicted values closely

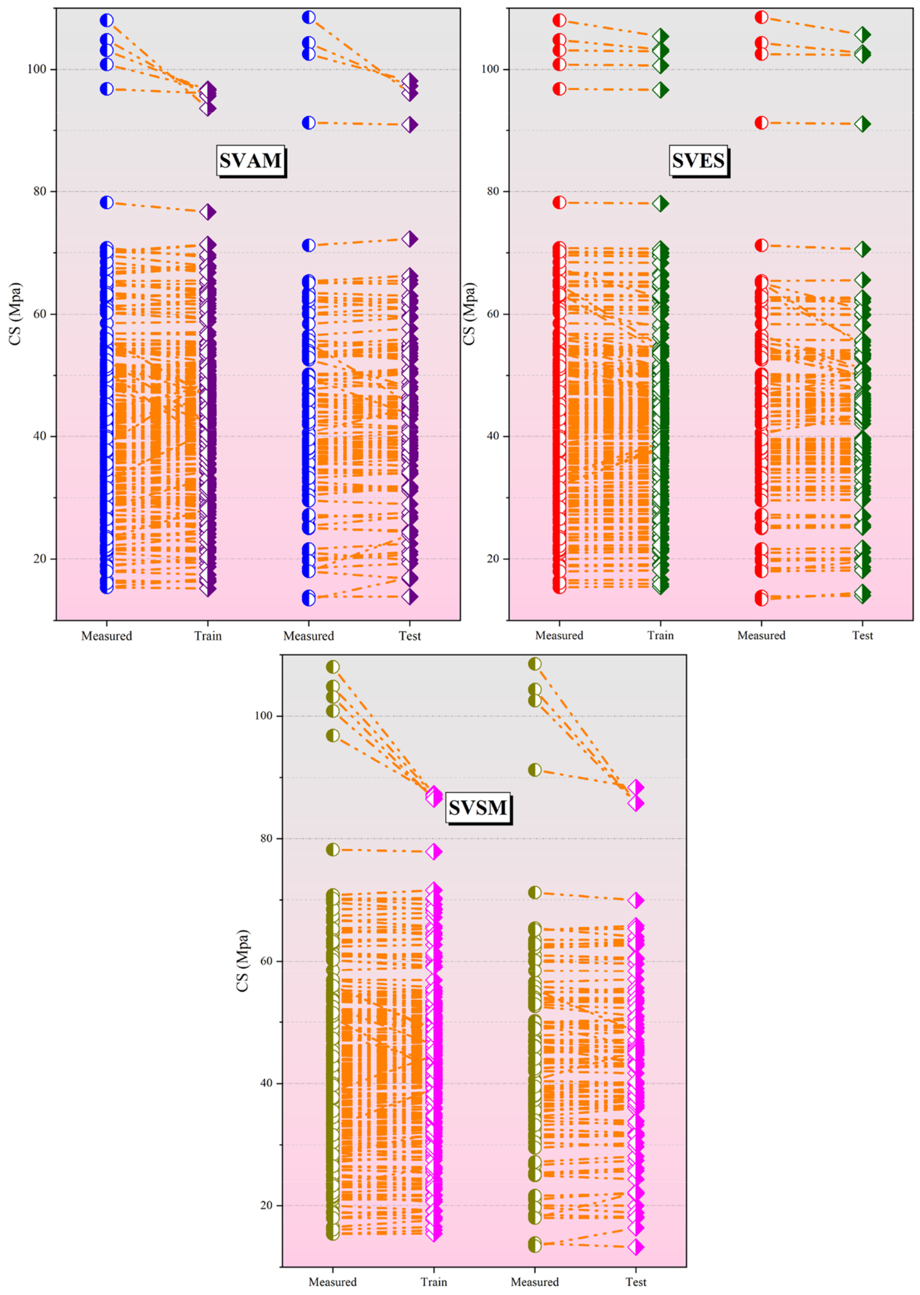

Fig. 4 The line series plot for presented hybrid models

following the measured values. However, there are some differences in performance among the models. SVES performs slightly more well than the other two models, as its predicted values are consistently closer to the measured values. SVAM and SVSM exhibit fluctuations in their predicted values, particularly at higher CS values. Overall, the line series plot provides a clear and concise visualization of the model's performance in estimating HPC strength and suggests that SVES may be the most accurate of the three models.

Figure 5 presents a graphical representation of the error percentage for the developed models during the training and test phases. The *x*-axis displays the models as well as the training/testing phases, while the *y*-axis indicates the error percentage. The violin plot for SVES illustrates the smallest range of error percentages, indicating that it is the most accurate of the three models. On the other hand, SVAM and SASM models demonstrate larger ranges of error percentages, suggesting that they are less precise than SVES. However, the violin plot also reveals that all three models display significantly lower error percentages during the testing phase than during the training phase. This suggests that the models may be overfitting to the training data and that their predictive capabilities may be limited when applied to new data. Overall, the graphical representation in Fig. 5 visually compares the models' error percentages and highlights the potential limitations of overfitting during the training phase.

Figure 6 presents a line and symbol plot that compares the performance of three models (SVAM, SVES, and SVSM) in estimating the CS of HPC in terms of error percentage. The *x*-axis depicts the sample number, while the *y*-axis depicts the error percentage. The plot illustrates that all three models are capable of estimating the HPC strength with a relatively low error percentage. However, there are differences in performance among the models. SVES exhibits the lowest error percentage, indicating that it is the most accurate model. SVAM and SVSM, on the other hand, display slightly higher error percentages, indicating that they are less precise than SVES.

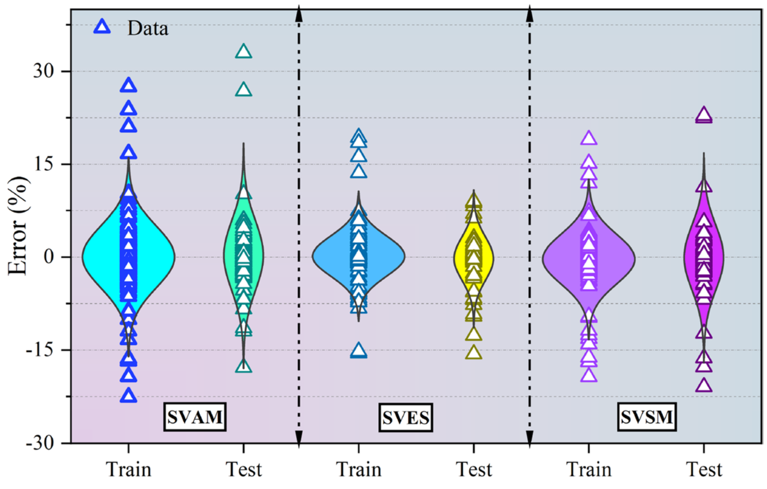

Fig. 5 The violin diagram for error% in testing and training of developed models

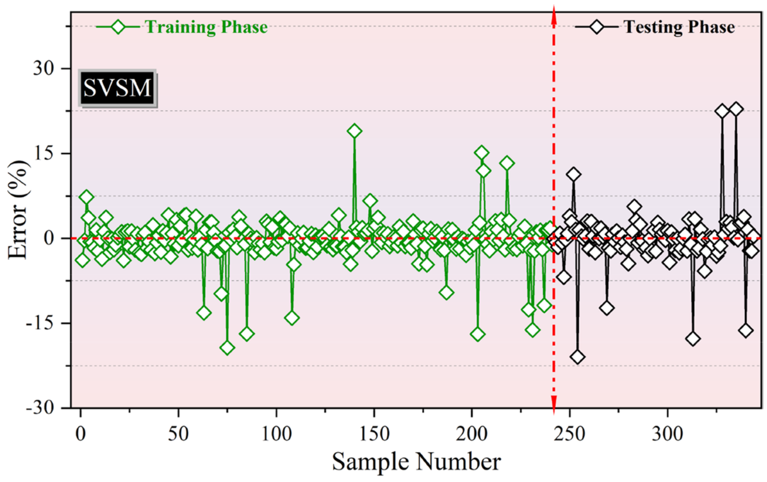

Fig. 6 The error percentage of the presented models based on line-symbol

Conclusions

In recent years, ML algorithms have become increasingly popular for estimating the CS of HPC. One such algorithm, the SVR model, has demonstrated potential in accurately predicting HPC strength. Nevertheless, the performance of the SVR model can be improved by selecting suitable optimization algorithms. In this investigation, the performance of the SVR model was compared with three optimization algorithms: slime mold algorithm (SMA), equilibrium slime mold algorithm (ESMA), and adaptive opposition slime mold algorithm (AOSMA). The findings revealed that all three optimization algorithms were successful in enhancing the SVR model's performance in predicting the compressive strength of HPC. Nevertheless, the ESMA algorithm performed the most, displaying the MAE and the highest R^2. The primary outcomes of the investigation are as follows:

(1) This investigation discovered that the SVR models, including SVES, SVSM, and SVAM, have a significant potential to predict CS, with a minimum R^2 value of 0.9740 during the training stage and 0.9687 during the testing phase. However, analyzing the distribution of data samples around the best-fit line revealed that the SVSM and SVAM models displayed lower performance in predicting CS values compared to the SVES model. It was observed that the ESMA optimization algorithm outperformed the other optimization algorithms.

(2) All statistical indices suggest that the performance of SVES is superior to the other SVR models with R^2, RMSE, WAPE, MRAE, and MAE values of 0.9894, 1.8396, 0.8506, 0.019, and 01752, respectively. On the other hand, the SVSM model displays the most inferior performance, with the lowest R^2, RMSE, WAPE, MRAE, and MAE values in both the testing and training phases.

In conclusion, combining the SVR model with the ESMA optimization algorithm can accurately forecast the compressive strength of HPC, which has significant implications for the construction industry regarding cost-effective and efficient construction processes. The ESMA algorithm shows great potential as an effective optimization algorithm in various engineering and scientific applications. Further research is necessary to investigate its potential in other optimization problems.

Advice for structural engineers

In navigating the dynamic landscape of construction and design, new opportunities for enhancing the accuracy and efficiency of predictions, particularly in the realm of HPC, are presented through the integration of machine learning algorithms and optimization techniques. The following advice is offered, drawing from the outcomes of this investigation:

(1) *Embrace ML Technologies*: Embrace the use of machine learning technologies, such as the SVR model, as powerful tools for predicting the CS of HPC. These technologies have shown significant potential in capturing complex relationships within concrete properties.
(2) *Optimize Model Performance*: Recognize the importance of optimization algorithms in refining the performance of ML models. The findings highlight that the choice of an optimization algorithm, with ESMA demonstrating superior results in this study, can substantially improve the accuracy of predictions.
(3) *Consider Model Variations*: When deploying SVR models, be mindful of variations such as SVES, SVSM, and SVAM. Understanding the strengths and limitations of each variation is crucial for selecting the most suitable model for specific applications.
(4) *Continuous Evaluation and Improvement*: Engage in a continuous process of evaluation and improvement. Regularly assess the performance of ML models against real-world data and be open to refining approaches based on evolving industry standards and technological advancements.
(5) *Explore ESMA in Other Applications*: Given the promising performance of the ESMA, consider exploring its potential in various engineering and scientific applications beyond concrete strength prediction. This algorithm may prove valuable in optimizing solutions for diverse optimization problems.
(6) *Collaborate and Share Knowledge*: Foster a culture of collaboration within the structural engineering community. Share knowledge and insights gained from the integration of ML algorithms, promoting a collective effort to advance the field and address emerging challenges.

Suggestions for future work

As this investigation is concluded, several avenues for future research emerge, providing opportunities to deepen the understanding and refine the application of machine learning in structural engineering. The following directions for future work are proposed:

(1) *Exploration of multifactorial influences*: Extend research efforts to explore the influence of additional factors on the predictive accuracy of machine learning models for concrete strength. Consider variables such as curing conditions, environmen-

tal factors, and mix design intricacies to create a more comprehensive predictive framework.

(2) *Integration of real-time data*: Investigate the feasibility of incorporating real-time data into machine learning models. The inclusion of up-to-the-minute information during the construction phase could enhance the adaptability and responsiveness of predictive models.

(3) *Long-term performance predictions*: Shift focus towards the long-term performance predictions of concrete structures. Evaluate the ability of machine learning models to anticipate changes in compressive strength over extended periods, considering factors like aging, environmental exposure, and structural loading.

(4) *Robustness under limited data conditions*: Explore the robustness of machine learning models, particularly SVR with optimization algorithms, under conditions of limited data availability. Develop strategies to enhance model performance when faced with sparse datasets, common in certain construction scenarios.

(5) *Incorporation of uncertainty analysis*: Integrate uncertainty analysis techniques into the predictive models to provide a more nuanced understanding of the confidence levels associated with predictions. This can contribute to more informed decision-making in practical engineering applications.

(6) *Implementation in industry practices*: Explore the practical implementation of machine learning models in real-world construction projects. Investigate the challenges and opportunities associated with integrating these models into existing industry practices, with a focus on scalability and usability.

Acknowledgements
I would like to take this opportunity to acknowledge that there are no individuals or organizations that require acknowledgment for their contributions to this work.

Authors' contributions
All authors contributed to the study's conception and design. Data collection, simulation, and analysis were performed by "Linlin Yu".

Funding
This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Availability of data and materials
Data can be shared upon request.

Declarations

Ethical approval and consent to partcipate
Research involving Human Participants and Animals: The observational study conducted on medical staff needs no ethical code. Therefore, the above study was not required to acquire an ethical code.
Informed consent: This option is not necessary due to that the data were collected from the references.

Competing interests
The author declares no competing interests.

Received: 14 October 2023 Accepted: 5 December 2023
Published online: 03 January 2024

References

1. Haile BF, Jin DW, Yang B, Park S, Lee H-K (2019) Multi-level homogenization for the prediction of the mechanical properties of ultra-high-performance concrete. Constr Build Mater 229:116797
2. Deepa C, SathiyaKumari K, Sudha VP (2010) Prediction of the Compressive Strength of High Performance Concrete Mix using Tree Based Modeling. Int J Comput Appl 6(5):18–24. https://doi.org/10.5120/1076-1406

3. Liu Y (2022) High-performance concrete strength prediction based on machine learning. Comput Intell Neurosci 2022
4. Li Q-F, Song Z-M (2022) High-performance concrete strength prediction based on ensemble learning. Constr Build Mater 324:126694
5. Zain MFM, Mahmud HB, Ilham A, Faizal M (2002) Prediction of splitting tensile strength of high-performance concrete. Cem Concr Res 32(8):1251–1258
6. Erdal HI (2013) Two-level and hybrid ensembles of decision trees for high performance concrete compressive strength prediction. Eng Appl Artif Intell 26(7):1689–1697
7. Wu X et al (2022) Prediction of the frost resistance of high-performance concrete based on RF-REF: A hybrid prediction approach. Constr Build Mater 333:127132
8. Liu Y, Cao Y, Wang L, Chen Z-S, Qin Y (2022) Prediction of the durability of high-performance concrete using an integrated RF-LSSVM model. Constr Build Mater 356:129232
9. 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
10. Chithra S, Kumar SRRS, Chinnaraju K, Ashmita FA (2016) A comparative study on the compressive strength prediction models for High Performance Concrete containing nano silica and copper slag using regression analysis and Artificial Neural Networks. Constr Build Mater 114:528–535
11. S. N. Mehdi Yaltaghian Khiabani1, Behnam sedaghat 2, Parisa Ghorbanzadeh3, Negin Porroustami4, Seied Mehdy Hashemy Shahdany5, Yousef Hassani6 (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
12. Han Q, Gui C, Xu J, Lacidogna G (2019) A generalized method to predict the compressive strength of high-performance concrete by improved random forest algorithm. Constr Build Mater 226:734–742. https://doi.org/10.1016/j.conbuildmat.2019.07.315
13. Cheng M-Y, Firdausi PM, Prayogo D (2014) High-performance concrete compressive strength prediction using Genetic Weighted Pyramid Operation Tree (GWPOT). Eng Appl Artif Intell 29:104–113. https://doi.org/10.1016/j.engappai.2013.11.014
14. 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(8):3487–3501. https://doi.org/10.2166/ws.2020.241
15. Ebid AM (2021) 35 Years of (AI) in geotechnical engineering: state of the art. Geotech Geol Eng 39(2):637–690
16. Akbarzadeh MR, Ghafourian H, Anvari A, Pourhanasa R, Nehdi ML (2023) Estimating Compressive Strength of Concrete Using Neural Electromagnetic Field Optimization. Materials (Basel) 16(11):4200
17 TavanaAmlashi A, MohammadiGolafshani E, Ebrahimi SA, Behnood A (2023) Estimation of the compressive strength of green concretes containing rice husk ash: a comparison of different machine learning approaches. Eur J Environ Civ Eng. 27(2):961–983. https://doi.org/10.1080/19648189.2022.2068657
18. Khajeh A, Ebrahimi SA, MolaAbasi H, JamshidiChenari R, Payan M (2021) Effect of EPS beads in lightening a typical zeolite and cement-treated sand. Bull Eng Geol Environ 80(11):8615–8632. https://doi.org/10.1007/s10064-021-02458-1
19. SarkhaniBenemaran R, Esmaeili-Falak M, Katebi H (2022) Physical and numerical modelling of pile-stabilised saturated layered slopes. Proc Inst Civ Eng Eng. 175(5):523–538
20. Benemaran RS, Esmaeili-Falak M (2020) Optimization of cost and mechanical properties of concrete with admixtures using MARS and PSO. Comput Concr 26(4):309–316. https://doi.org/10.12989/cac.2020.26.4.309
21. 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 1–20
22. Vapnik V, Golowich S, Smola A (1996) Support vector method for function approximation, regression estimation and signal processing. Adv Neural Inf Process Syst 9
23. Hameed MM, AlOmar MK (2020) Prediction of compressive strength of high-performance concrete: hybrid artificial intelligence technique," in Applied Computing to Support Industry: Innovation and Technology: First International Conference, ACRIT 2019, Ramadi, Iraq, September 15–16, 2019, Revised Selected Papers 1. 323–335
24. Sedaghat B, Tejani GG, Kumar S (2023) Predict the Maximum Dry Density of soil based on Individual and Hybrid Methods of Machine Learning. Adv Eng Intell Syst 2;(03). https://doi.org/10.22034/aeis.2023.414188.1129
25. Yu Y, Li W, Li J, Nguyen TN (2018) A novel optimised self-learning method for compressive strength prediction of high performance concrete. Constr Build Mater 184:229–247
26. Ahmad A, Chaiyasarn K, Farooq F, Ahmad W, Suparp S, Aslam F (2021) Compressive strength prediction via gene expression programming (GEP) and artificial neural network (ANN) for concrete containing RCA. Buildings 11(8):324
27. Zhang F, O'Donnell LJ (2020) Support vector regression. In: Machine learning. New York: Elsevier. pp. 123–140
28. Gunn SR (1998) Support vector machines for classification and regression. ISIS Tech Rep 14(1):5–16
29. Li L-L, Chang Y-B, Tseng M-L, Liu J-Q, Lim MK (2020) Wind power prediction using a novel model on wavelet decomposition-support vector machines-improved atomic search algorithm. J Clean Prod 270:121817
30. Luenberger DG, Ye Y (1984) Linear and nonlinear programming, vol. 2. Amsterdam: Springer
31. Li S, Chen H, Wang M, Heidari AA, Mirjalili S (2020) Slime mould algorithm: A new method for stochastic optimization. Futur Gener Comput Syst 111:300–323
32. Abdel-Basset M, Chang V, Mohamed R (2020) HSMA_WOA: A hybrid novel Slime mould algorithm with whale optimization algorithm for tackling the image segmentation problem of chest X-ray images. Appl Soft Comput 95:106642

33. Chen H, Li C, Mafarja M, Heidari AA, Chen Y, Cai Z (2023) Slime mould algorithm: a comprehensive review of recent variants and applications. Int J Syst Sci 54(1):204–235
34. Yin S, Luo Q, Zhou Y (2022) EOSMA: an equilibrium optimizer slime mould algorithm for engineering design problems. Arab J Sci Eng 47(8):10115–10146
35. Naik MK, Panda R, Abraham A (2021) An entropy minimization based multilevel colour thresholding technique for analysis of breast thermograms using equilibrium slime mould algorithm. Appl Soft Comput 113:107955
36. Altay O (2022) Chaotic slime mould optimization algorithm for global optimization. Artif Intell Rev 55(5):3979–4040