Case Studies in Construction Materials 18 (2023) e01928

Contents lists available at ScienceDirect

Case Studies in Construction Materials

journal homepage: www.elsevier.com/locate/cscm

Case study

Efficient boosting-based algorithms for shear strength prediction of squat RC walls

Alireza Farzinpour [a], Esmaeil Mohammadi Dehcheshmeh [a], Vahid Broujerdian [a,*], Samira Nasr Esfahani [b], Amir H. Gandomi [c,d,**]

[a] School of Civil Engineering, Iran University of Science and Technology, Tehran, Iran
[b] Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran
[c] Faculty of Engineering & IT, University of Technology Sydney, Sydney, NSW, Australia
[d] University Research and Innovation Center (EKIK), Óbuda University, 1034 Budapest, Hungary

ARTICLE INFO

Keywords:
Squat RC wall
Genetic algorithm (GA)
Hyperparameter optimization
Boosting methods
Principal component analysis (PCA)
Machine learning

ABSTRACT

Reinforced concrete shear walls have been considered as an effective structural system due to their optimal cost and great behavior in resisting lateral loads. For the slender type of these walls, failure modes are mainly related to flexure, while for the squat type with height-to-length ratios less than two, shear is the dominant factor. Thus, accurate estimation of shear strength for squat shear walls is necessary for design applications and can also be complex due to the various effective parameters. In order to address this issue, first a comprehensive dataset with 558 samples of squat shear walls is conducted, and three hybrid models consisting of genetic algorithms and boosting-based ensemble learning methods, i.e., XGBoost, CatBoost, and LightGBM, are used for estimation of shear strength. The results showed high prediction accuracy, with a coefficient of determination of at least 98.6% for all three models. Genetic algorithm has been proven to be an effective method for tuning boosting-based algorithms compared to manual testing. In addition, the results of the algorithms are compared to their default hyperparameters and other conventional regression Models. Also, multicollinearity and principal component analysis (PCA) were studied. Furthermore, the performance of three tuned models is compared with that of a mechanics-based semi-empirical model and other genetic programming (GP)-based models. Finally, parametric and sensitivity analyses were performed, to demonstrate the ability of the models to identify the most critical parameters with significant influence on shear strength.

1. Introduction

Reinforced Concrete (RC) shear walls are frequently used in tall structures as a lateral load resisting system. As a result, in seismic analysis and design processes [1] and [2], precise capacity predictions of these systems are a critical factor. There are two main characteristics that difficult these predictions: flexure shear and concrete–reinforcement interactions. This task becomes design-critical, especially for squat shear walls (that have height-length ratios of less than two), as shear force, when compared with flexure, is significantly more likely to break these walls. Attempts to build mechanics-based shear strength models for squat walls, such

* Correspondence to: Iran University of Science and Technology, Narmak, Tehran, Iran.
** Correspondence to: University of Technology Sydney, 15 Broadway, Ultimo, NSW 2007, Australia.
E-mail addresses: broujerdian@iust.ac.ir (V. Broujerdian), gandomi@uts.edu.au (A.H. Gandomi).

https://doi.org/10.1016/j.cscm.2023.e01928
Received 19 October 2022; Received in revised form 29 January 2023; Accepted 8 February 2023
Available online 10 February 2023

as the strut and tie model, have been made in recent decades [3] and [4], or the softened truss model [5] and [6]. Because these models simplify the complex nonlinear behavior of concrete, their estimates have some degree of scattering (and some bias) [7] and [8].

Machine learning (ML) algorithms, as well as deep learning algorithms, are widely employed in many engineering sectors and have broadened the scope of structural and seismic engineering studies. Due to the increasing development of precise and fast ML algorithms, as well as the availability of enough accurate experimental data, using these techniques to estimate and interpret the limited state-based criteria associated with design codes has become acceptable by the community. Several studies have been done using these algorithms in structural engineering and optimization phases in the recent years [9].

Mangalathu and Jeon [10] developed a methodology with an artificial neural network, and tested it with a dataset based on experiments for reinforced columns with circular sections. This method was able to predict how the columns will fail (shear, flexure, or flexure-shear). A similar study was carried out on shear walls to discover the failure process, which is essential to the current endeavor [11]. Siam et al. [12] classified and predicted the performance of 97 reinforced masonry shear wall samples using ML models. Feng et al. [13] used an ensemble learning approach to categorize the failure mode of RC columns, as well as predict the maximum lateral force (V_{max}) in the load-displacement curve of columns. They also used the same ML technique to predict column plastic hinge length, which is a critical parameter in time history analysis for performance-based seismic assessment [14]. Thinh Le and Vuong [15] used an ML model based on Gaussian regression for predicting the load-carrying capacity of filled steel tubular (CFST) columns. Their results were also validated with other ML models like Artificial neural network, Support vector machine and several other code formulations (such as, EC4, AISC and ACI) for estimating load capacity of these columns.

Wu and Zhou [16] used a combination of support vector regression and grid search optimization as a hybrid approach to predict the compressive strength of sustainable concrete. They also used the Shapley additive explanation (SHAP) method to explain the importance of features on compressive strength. Feng et al. [17] used the same approach for shear strength of squat shear walls with XGBoost for prediction and SHAP algorithm for interpretation. An ML technique that uses Genetic Programming (GP) was reported to estimate the shear strength of squat walls, at the same time it applies specific mechanics-guided inferences to develop an unambiguous expression for shear strength, which resulted in a model with great accuracy and practicality [18].

Using multi-expression programming (MEP), Gandomi et al. [19] developed new design equations to assess the shear resistance of steel fiber-reinforced concrete beams (SFRCB). The ability of MEP to model mechanical phenomena without the requirement to predefine the model structure distinguishes it from traditional statistical methodologies. Also, Aravind et al. [20] employed six ML algorithms for classifying and detecting failure types of geopolymer concrete beams.

ML methods are appropriate for a wide range of issues and datasets. Finding the most appropriate and best ML model architecture is an iterative and time-consuming procedure [21]. This task is especially vital for hyperparameters which cannot be directly obtained from pattern learning and must be set before the learning phase [22]. Hyperparameters in the ML area are used for two main purposes. First, they are used to define the structure of a ML model (such as several hidden layers and units in neural networks). Second, they are used to specify the technique used to minimize the loss function in the optimization process of learning (for example, learning rate and the number of epochs in neural networks) [23]. Choosing the best configuration of hyperparameters is crucial to find the ideal performance of a model, being known as hyperparameter tuning. Tuning hyperparameters is an important step when developing a good ML model, especially for models which have a lot of these parameters [24]. Manual testing is an old approach for tuning the hyperparameters, with the drawback that it requires a deep understanding of hyperparameter values, as well as their behavior on algorithm performance [25]. Also, manual tuning can be ineffective for algorithms with an ample hyperparameter search space, an intricate structure, and in the presence of nonlinear hyperparameter interactions. These factors have motivated increased research into hyperparameter optimization (HPO) techniques, whose primary purpose is to automate the tuning process and enable users to effectively apply ML algorithms. After an HPO procedure, the best-performed ML algorithm structure for the considered problem should be achieved [21] and [26]. Some of the most compelling reasons to employ HPO techniques on ML models are listed below [24]:

1) Since many ML developers spend a substantial amount of time optimizing hyperparameters, this will decrease the amount of human effort necessary, especially for tasks with many features or complicated algorithms with an ample search space for hyperparameters.
2) It helps ML models to perform better. When changing datasets or applications, different ML hyperparameters have different optimum values.
3) It helps making models more easily repeatable. Multiple ML algorithms can only be adequately compared when the same degree of hyperparameter tuning strategy is employed. Thus, applying the same HPO approach to various ML algorithms aids in selecting the optimal ML model for a given problem.

Metaheuristic algorithms, which are collections of strategies for addressing complex, high search space, and non-convex optimization problems, can be used to tackle HPO problems [27]. Among all metaheuristic approaches, genetic algorithm (GA) [28] and particle swarm optimization (PSO) [29], are the two most common algorithms used for HPO problems. GAs uncover high-performing hyperparameter combinations in each generation, passing them on to the next generation, and until the best performing combination is discovered [30]. In PSO algorithms, each particle communicates with other particles to find and update the current global optimum in each iteration, until the ultimate optimum is found. Metaheuristics can quickly explore the search space for the best or near-best solutions [29] and therefore, their excellent efficiency is particularly well suited to HPO issues with huge configuration spaces [30]. For example, they can be used in tree-based algorithms with a vast configuration space and many hyperparameters [30]. In tree-based methods, boosting algorithms are widely used because of their high accuracy and fast performance. Furthermore,

combining them with a metaheuristic approach can increase the accuracy and speed of both the hyperparameter tuning and prediction phases. Different works also have been employed regarding these matters. For example, Mohit Jane [31] used a GA to tune the XGBoost hyperparameters for classification problems. Telikani et al. [32] investigated evolutionary computation approaches to improve the performance of ML models in phases like preprocessing, learning and postprocessing. Zhang et al. examined a modified type of beetle antennae search (BAS) algorithm for hyperparameter tuning the random forest model in order to predict the shear strength of concrete beams based on two types of datasets for beams with and without stirrups [33]. Sun et al. proposed a 360-sample dataset for coalcrete material and used support vector machines (SVM) optimized by the BAS algorithm for predicting the young modulus of the material [34]. In another study, the evaluation of PSO-tuned ensemble algorithms is considered for the strength assessment of jet-grouted coal-grout composites [35].

In this paper, First, a new dataset of 558 samples is assembled from the literature. Then, the effectiveness of the GA for hyperparameter optimization of various boosting-based methods, such as XGBoost, CatBoost, and LightGBM, will be explored on the proposed dataset for shear strength prediction. In order to make a comparison, consideration of these three boosting-based algorithms with their default hyperparameters is also examined, alongside other conventional ML methods such as random forest (RF), decision trees (DT), artificial neural network (ANN), and support vector machine (SVM). Models' performances are evaluated based on five measurement metrics. Also, the correlation matrix of the dataset is extracted, and the presence of multicollinearity is studied. Principal component analysis (PCA) as a way to tackle multicollinearity is performed on the dataset, with four and nine principal components with at least 72% and 95% informative performance of the primary dataset being selected and considered as two new datasets. After that, performance of GA-boosting models on the datasets given dimension reduction is compared based on the same five metrics. In the third step, the predictions from the GA are tuned into ML models, being compared with a mechanics-based and GP extracted [18] models. In the end, sensitivity and parametric analysis [36] are conducted, input features are compared based on their influence on the

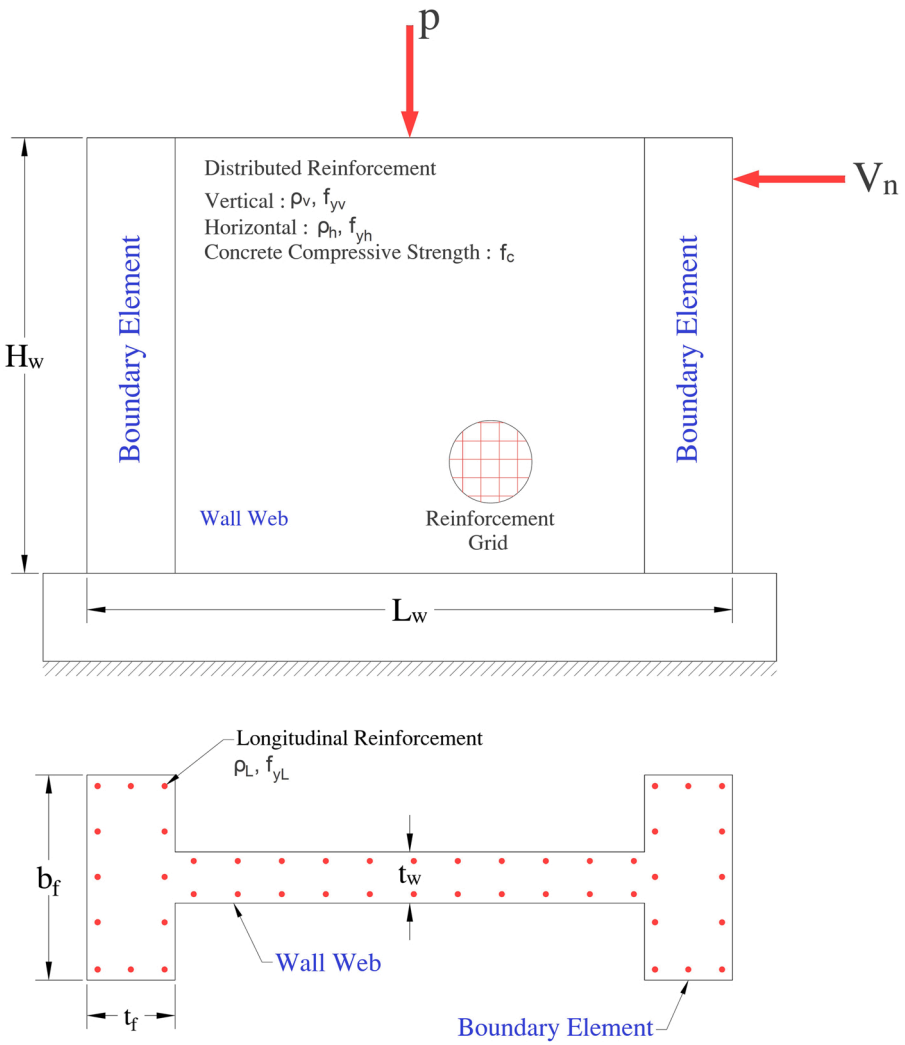

Fig. 1. Schematic diagram of squat RC walls.

resulted estimations, and conclusions are made according to the results of the different steps.

2. Material and methods

2.1. Squat shear wall data set

An experimental dataset is required to create an adequate estimation model for predicting the shear strength of squat shear walls. A new set of 558 squat shear wall tests has been compiled, which combines earlier datasets from Ma et al. [37]- [38] and Feng et al. [17]. This set can improve the prediction accuracy of ML models because of the large number of test instances. The schematic depiction of the squat RC wall tests in the dataset is shown in Fig. 1. The four input category features are the geometric dimensions, reinforcement ratio, material characteristics, and applied loads, as depicted in this diagram, and Table 1 shows a statistical description of the input features.

2.2. Genetic algorithm

GA [28] is a popular metaheuristic and population-based optimization algorithm(POAs) [39] and [40] based on natural selection and genetics. More specifically on the fact that people with the highest survival and environmental adaption have more chances to live and pass on their qualities to next generations. Various studies have been done based on the use of POAS in different fields, like structural engineering [41] or hyperparameter optimization [28], [29], and [42]. In GA, each chromosome, or person, represents a hyperparameter, and its decimal value is the actual input value for the hyperparameter in each evaluation. Every chromosome has several binary digit genes prone to crossover and mutation activities. The population comprises all possible values within the initialized chromosome/parameter ranges, whereas the fitness function describes the population's fitness [43]. The following are the main GA procedures [39]:

1) Initialize the population, chromosomes, and genes randomly, representing the whole search space, hyperparameters, and hyperparameter values, respectively.
2) To evaluate the performance of each individual in the current generation, calculate the optimal function, which is the objective function of an ML model.
3) To create a new generation with the next set of hyperparameter configurations to be evaluated, perform chromosomal selection, crossover, and mutation operations.
4) Repeat steps 2 and 3 until the termination condition is satisfied.
5) Terminate and output the ideal hyperparameter configuration.

The population initialization phase in GA and PSO is crucial, since it provides an initial estimate of the optimum values. Near-global optimum solutions should be included in a good initial population of hyperparameters, by covering the prospective areas and not being restricted to an unpromising part of the search space [44]. Random initialization, which essentially provides an initial population with random values in the search space, is frequently used in GA to generate hyperparameter configuration possibilities for the first population [45]. GA has $O(n^2)$ time complexity, and as a result, GA might be inefficient due to its low convergence speed. The fundamental issue of GA is that it requires the user to define extra hyperparameters, such as the fitness function type, population size, crossover rate, and mutation rate. Furthermore, and since GA is a sequential execution technique, parallelization is challenging [46].

2.3. Extreme gradient boosting (XGBoost)

XGBoost is a type of ensemble learning approaches, which are used to predict and interpret the mechanical behavior of concrete structures in some researches [17] and [47]. Compared to the gradient boosted decision tree (GBDT), XGBoost has improvements in

Table 1
Statistical description of dataset input features.

Features/value	mean	std	min	25%	50%	75%	max
Hw	918.91	516.02	145.00	500.00	860.00	1300.00	2200.00
Lw	1328.24	769.94	420.00	600.00	1180.00	1905.00	3960.00
tw	76.09	43.89	10.00	40.00	70.00	101.60	160.00
fc	28.56	14.52	12.30	19.10	25.50	33.00	104.00
ρv	0.0070	0.0054	0	0.0036	0.0056	0.0090	0.0367
fyv	367.86	114.84	0	303.26	369.00	433.00	624.00
ρh	0.0069	0.0050	0	0.0034	0.0057	0.0092	0.0367
fyh	368.70	115.49	0	302.30	369.00	433.00	624.00
ρL	0.0303	0.0202	0.0035	0.0150	0.0250	0.0461	0.1058
fyL	382.92	82.07	208.90	312.30	382.20	443.40	605.00
P	286.06	488.90	0	0	0	448.25	2365.00
tf	118.29	69.70	30.00	80.00	101.60	150.00	360.00
bf	295.18	390.13	30.00	100.00	152.40	360.00	3045.00

multithreaded processing, the classifier, and optimization function. This algorithm controls the complexity of the tree and reduces overfitting by adding a regularization term to the objective function. A column sampling technique is employed to prevent overfitting. The second order Taylor expression of the objective function is used to make the definition of the objective function simpler and more precise when finding the optimal solution [17] and [48].

Considering $D\{(x_i, y_i)\}$ as a dataset with n samples and m features, the predictive variable is an additive model which is made up of k basic models and can be formulated as follow:

$$\widehat{y}_i = \phi(x_i) = \sum_{k=1}^{K} \alpha_k f_k(x_i) \tag{1}$$

where $\widehat{y}_i$ is the prediction value; $\phi()$ is the final strong learner; $f_k()$ is the weak learner (the decision tree (DT) technique produces a weak learner); K is the number of weak learners; and α_k is the learning rate (to avoid overfitting, the learning rate was employed).

As mentioned earlier the objective function of XGBoost includes a regularization term which represent the complexity of the model and a traditional term for the loss between predicted and real values; as follow [17] and [48]:

$$Obj = \sum_{i=1}^{m} (y_i - \widehat{y}_i)^2 + \sum_{k} \Omega(f_k) \tag{2}$$

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w_k\| \tag{3}$$

where w_k is the leaf scores (or weights); and γ and λ are the penalty coefficients. Regularization term can smooth the final learning weight and avoid overfitting.

2.4. CatBoost

CatBoost is an improvement of GBDT algorithm similar to XGBoost. CatBoost provides a novel and effective method of dealing with categorical features throughout the learning process, using Order boosting to correct prediction shift problems and improve accuracy. The CatBoost method was intended to make it easier to deal with the category features in GBDT. CatBoost uses a more efficient method that avoids overfitting at the same time it allows for training on the entire dataset, which is achieved by randomly permuting the dataset and computing the average label value for each sample with the same category value placed before the given one in the permutation [42].

2.5. LightGBM

LightGBM is a Microsoft-published enhancement framework based on the decision tree method introduced in 2017 [49] and [50]. The significant features of LightGBM are to include a decision tree strategy based on gradient-based one-side sampling (GOSS), exclusive feature bundling (EFB), and a histogram and leaf-wise growth approach with a depth limit and unlike XGBoost, LightGBM would grow the tree vertically whereas other algorithms grow trees horizontally, which makes LightGBM an effective method in processing large-scale data and features [49] and [50]. Again considering $D\{(x_i, y_i)\}$ as a dataset with n samples, LightGBM objective function can be written as:

$$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \widehat{y}_i^t) + \sum_{i=1}^{t} \Omega(f_i) = \sum_{i=1}^{n} l(y_i, \widehat{y}_i^{(t-1)} + f_t(x_i)) + \sum_{i=1}^{t} \Omega(f_i) \tag{4}$$

With considering logistic loss and the Taylor expansion the objective function will be:

$$Obj^{(t)} = \sum_{i=1}^{n} \left[l\left(y_i, \widehat{y}^{(t-1)}\right) + g_i f_i(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{5}$$

where g_i and h_i denote the first- and second-order gradient statistics of the loss function.

Using the accumulation of n samples to pass over all of the leaf nodes as in Eq. 6, it yields:

$$Obj^{(t)} \cong \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{6}$$

Considering I_j denote the sample set of leaf j, the objective function could be transformed as follow:

$$Obj^{(t)} = \sum_{i=1}^{n} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \epsilon I_i} h_j + \lambda \right) w_j^2 \right] \tag{7}$$

For a certain tree structure, the partial derivative of the *jth* leaf node's output W_j is calculated, and the extreme value of objective function could be solved as follows:

$$\omega_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\epsilon I_i} h_j + \lambda} \tag{8}$$

$$L_t(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\sum_{i\in I_j} g_i{}^2}{\sum_{i\epsilon I_i} h_j + \lambda} \tag{9}$$

Finally, the objective function after adding split can be written as:

$$G = \frac{1}{2}\left[\frac{\sum_{i\in I_L} g_i{}^2}{\sum_{i\epsilon I_L} h_i + \lambda} + \frac{\sum_{i\in I_R} g_i{}^2}{\sum_{i\epsilon I_R} h_i + \lambda} - \frac{\sum_{i\in I} g_i{}^2}{\sum_{i\epsilon I} h_j + \lambda}\right] \tag{10}$$

where I_L and I_R are the sample sets of the left and right branches, respectively [50].

2.6. Multicollinearity and principal component analysis (PCA)

Multicollinearity refers to the state where independent variables in the dataset exhibit a strong relationship with each other. As a result, this can cause problems when the model is fitted, and interpretations are needed for the results. There are several methods to identify and tackle multicollinearity. Pearson's correlation coefficient metric directly evaluates the strength of the relationship between two variables, where its values range between − 1 and 1. The correlation coefficient's magnitude represents the relationship's strength, with a higher value corresponding to a stronger relationship. By calculating the correlation between pairs of predictive features, the presence of multicollinearity between them can be identified. Fig. 2 shows Pearson's correlation coefficient matrix.

The correlation matrix indicates that there are four pairs of features ((Hw, tw), (Lw, tf), (ρh, ρv), (fyh, fyv)) that indicate the presence of multicollinearity, due to correlation coefficient values of more than 0.7. One way to deal with multicollinearity is to use dimensionality reduction techniques, such as PCA analysis. PCA is a multivariate method used to evaluate a dataset where many intercorrelated quantitative dependent variables represent observations. Its purpose is to extract the essential information from the dataset, represent it as a set of new orthogonal variables called principal components, and display the similarity pattern of the observations and variables as dots on maps. Cross-validation techniques, such as the bootstrap or the jackknife, can be used to test the validity of the PCA model [51].

2.7. Model implementation

The original dataset is randomly divided into training and testing datasets for each of the three model implementations, with a division of 75% and 25% for training and testing sets, respectively. The GA steps for each model are as follows [31]:

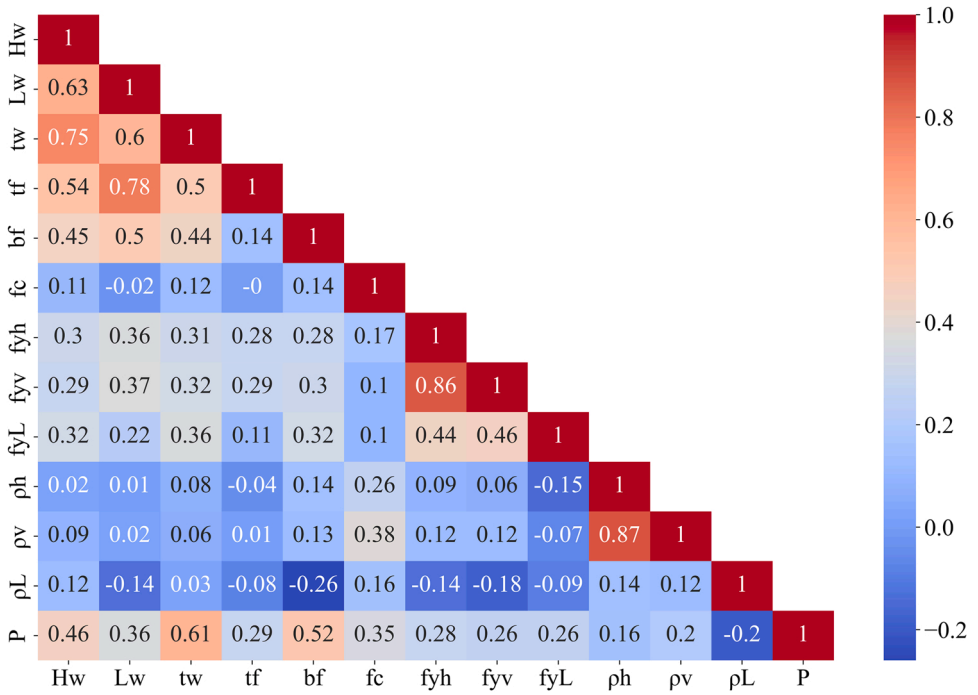

Fig. 2. Correlation matrix of a dataset.

1) Initialization, in which the parameters are randomly initialized to form the population.
2) The model is trained using the initial population and determining the fitness value. In the case of a regression problem, the cross-validation score on the coefficient of determination (R^2) is used.
3) Specify the number of parents to be chosen and build an array of the chosen parents based on their fitness value.
4) Use uniform crossover, in which each parameter for the child is chosen independently from the parents, based on a predefined distribution.
5) Genetic mutation, where variety is introduced to the children, by randomly picking one of the factors and changing its value by a random amount. Certain constraints are also added to keep the changed values within a specified range.

The performance of the three models is assessed using the training and testing sets, using the coefficient of determination (R^2), root-mean-square error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). Also, the performance index (PI), as proposed by Gandomi and Roke [52] is calculated. It is a function of both R and RMSE and its lower values (closer to zero) indicate better model performance. Its formula can be shown below:

$$PI = \frac{RRMSE}{R+1} \tag{11}$$

After that, PCA analysis is done on the data set, and a certain number of features are chosen based on the cumulative variance chart. Genetic-based techniques are used on the dimensionality reduction data set, and the results are investigated for the reduced dataset. Compression of models was made with a semi-empirical mechanics- and a GP-based model. In the end, sensitivity and parametric analysis are also conducted.

3. Results and discussions

3.1. Results on the basic dataset

The shear strength values for squat walls predicted by the genetically tuned XGBoost, CatBoost, LightGBM and other mentioned models for the entire dataset are described in this section. Table 2 presents the evaluation results of the models on the original dataset and for five accuracy metrics. According to four of five metrics, algorithms that are optimized with GA outperformed examinations with their default hyperparameter values on the test set. Also, GA-XGBoost outperformed all other conventional ML models. The RF and DT parameters are set to their default values. For ANN, 10 hidden layers are considered, with 4000 maximum iterations. And, for SVR, the C and gamma parameters are equal to 4000 and 0.2, respectively.

Figs. 3–6a illustrate the change in the values of the different hyperparameters in the population, during the generations of the GA, for the XGBoost model. Fig. 6**b** shows the R^2-score changes in the 10-fold cross-validation score on training sets, which is part of the fitness function in GA. Twelve parents are considered and updated in the following generations through the use of mutation and crossover methods. The algorithm started with a high fitness value of nearly 0.967 in the first parent at the first generation in the randomly initialized population, but it could improve in subsequent generations until the final generation of the first parent, where it reaches the highest value of 0.984. They suggest the value of each parameter for the best R2 performance, as seen in this generation of first parent from hyperparameter figures, with learning_rate = 0.203, n_estimator = 900, maximum depth = 14.0, min child weight = 10.0, subsample = 0.727, col sample by tree = 0.897 and reglambda = 7.201.

Table 2
Evaluation of different metrics on machine learning models.

Methods	Sets	R^2	RMSE (kN)	MAE (kN)	MAPE (%)	PI
GA-XGBoost	Training	0.9998	2.98	1.26	0.64	0.002
	Testing	0.9906	71.38	43.89	11.44	0.036
XGBoost (Default)	Training	0.9998	2.70	1.22	0.82	0.001
	Testing	0.9846	91.10	53.63	10.91	0.047
GA-CatBoost	Training	0.9995	14.69	10.46	5.54	0.008
	Testing	0.9861	86.53	50.29	14.91	0.044
CatBoost (Default)	Training	0.9994	16.26	11.79	5.80	0.009
	Testing	0.9852	89.46	51.67	12.10	0.047
GA-LightGBM	Training	0.9999	7.06	3.90	1.75	0.004
	Testing	0.9867	84.74	49.13	13.95	0.043
LightGBM (Default)	Training	0.9934	56.18	30.11	7.79	0.031
	Testing	0.9844	91.87	58.60	13.12	0.047
ANN	Training	0.9934	56.07	39.17	17.52	0.031
	Testing	0.9554	155.19	83.78	23.41	0.079
RF	Training	0.9960	43.74	22.79	4.98	0.024
	Testing	0.9851	89.64	56.19	11.59	0.046
DT	Training	0.9999	2.41	0.42	0.31	0.001
	Testing	0.9789	106.79	52.60	12.15	0.054
SVM	Training	0.9996	13.74	3.92	1.84	0.007
	Testing	0.9863	85.92	48.39	13.46	0.044

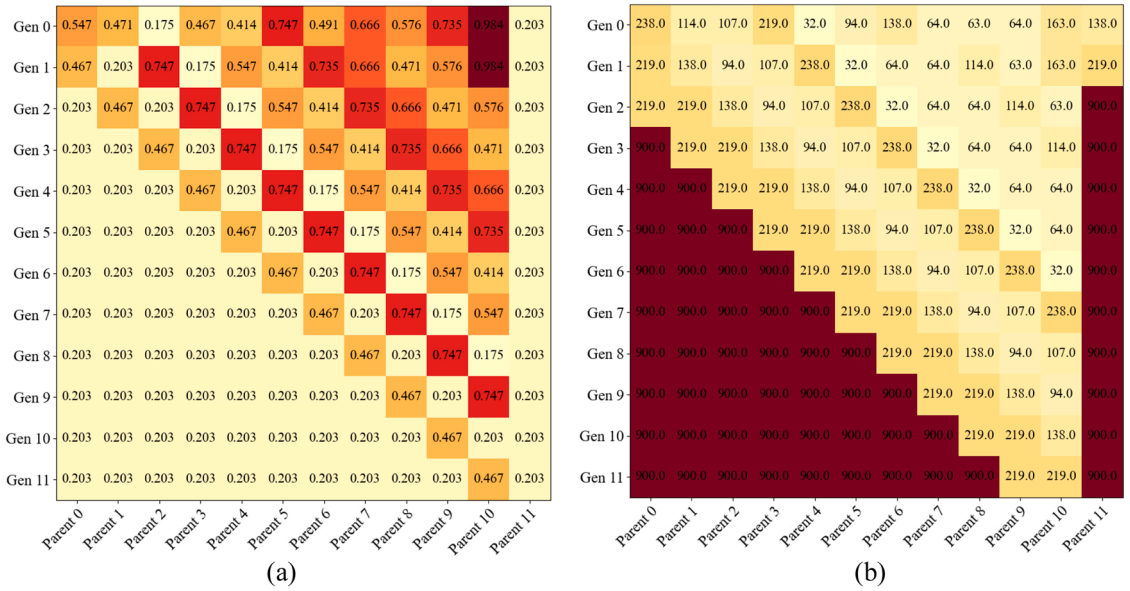

Fig. 3. Hyperparameter values history: (a) learning rate, and (b) n_estimator.

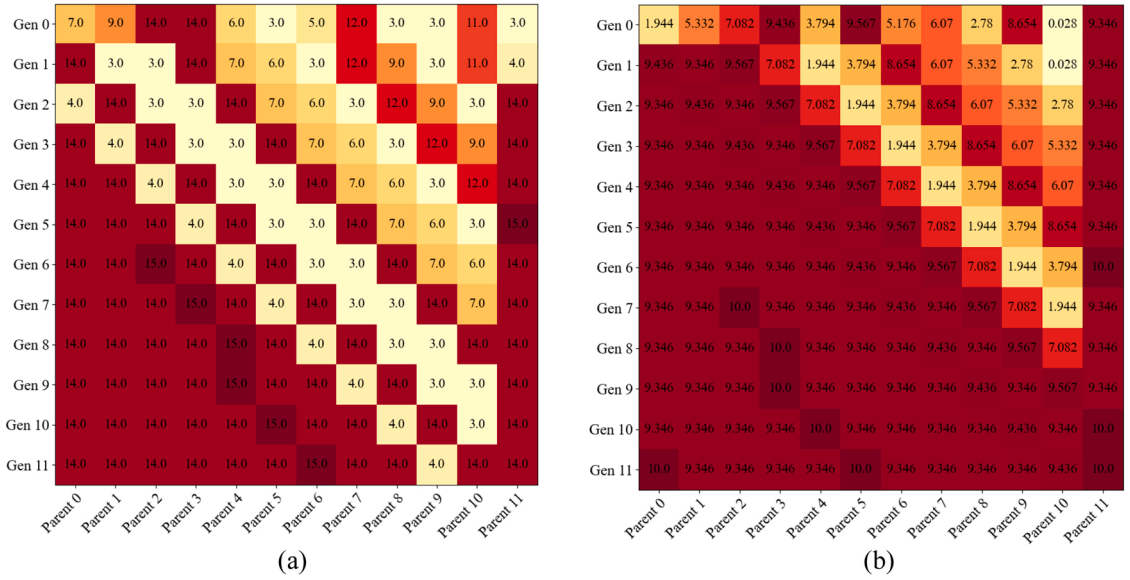

Fig. 4. Hyperparameter values history: (a) maximum depth, and (b) minimum child weight.

The total execution times for the training and testing phases of boosting-based algorithms and tuning the hyperparameters of methods with the mentioned genetic algorithm are shown in Table 3. According to the results, CatBoost had the best performance in both phases. However, results for the other two methods are also great, which shows the efficiency of boosting algorithms for prediction and metaheuristic methods for hyperparameter tuning. It should be noted that results can vary for datasets with different amounts of data and different genetic algorithm configurations.

In order to make a comparison between predictions and their actual values, Fig. 7 illustrates the measured shear strengths versus the model's predicted shear strengths for both training and testing sets, showing that shear strength can be accurately predicted using these models. The variation of scatters around the ideal y = x line is so low, and the prediction results of algorithms are closely reaching the line, implying that the predicted and tested values are almost identical. Also, among the three algorithms, GA-XGBoost performed the best with the lowest variation of scatters.

Fig. 8 displays the prediction-to-test ratios of shear strengths for the proposed dataset versus the aspect ratio of walls (i.e., hw/lw) from 0.0 to 2.0. The resulted mean ± standard deviation prediction intervals of ratio points are also provided as a result. As can be seen

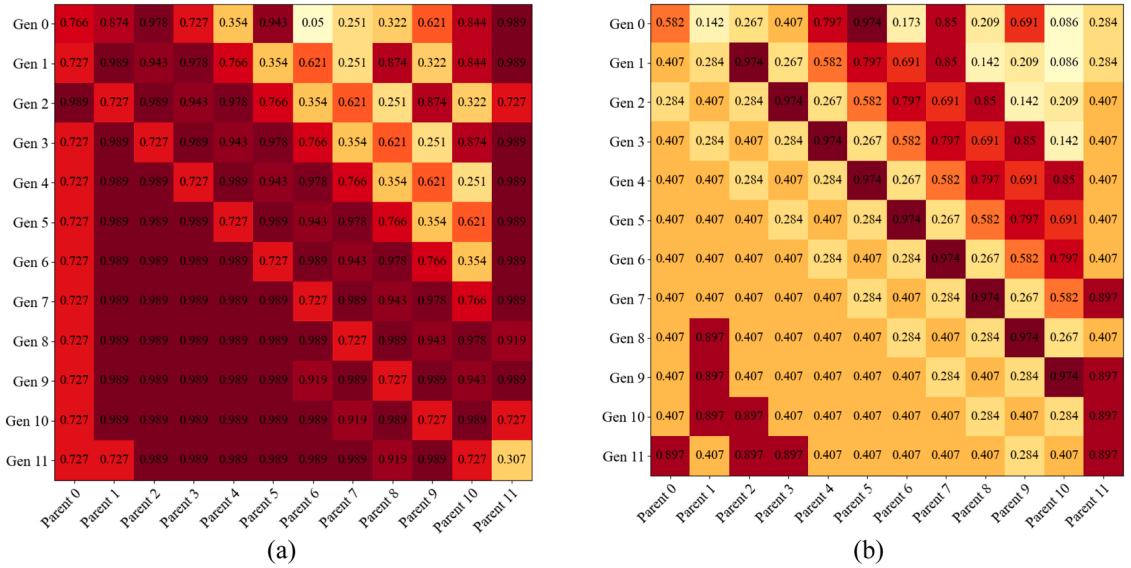

Fig. 5. Hyperparameter values history: (a) subsample, and (b) col sample by tree.

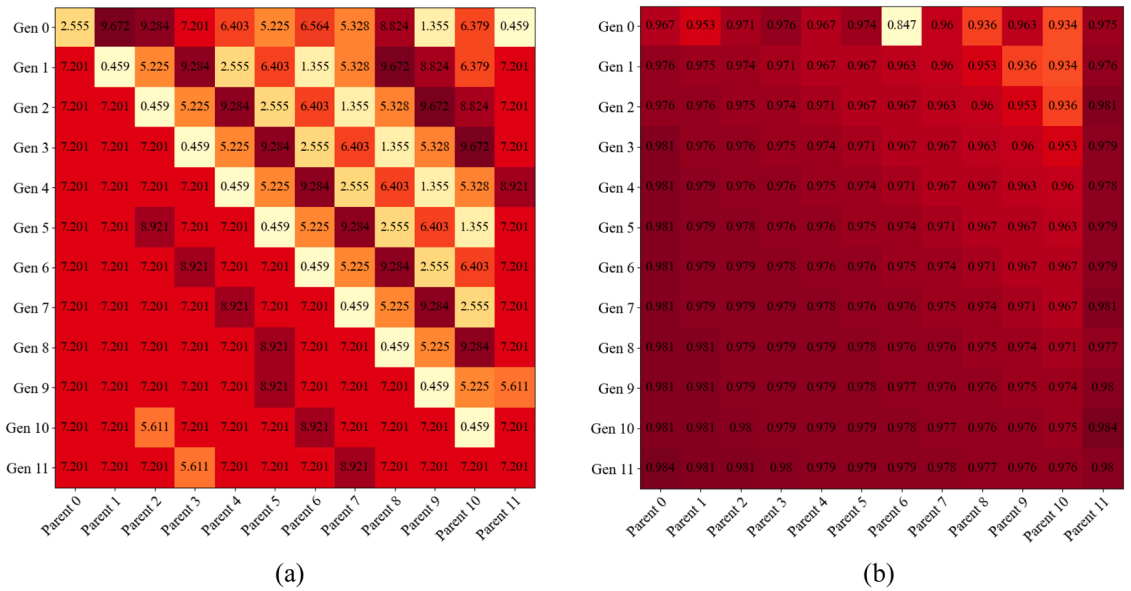

Fig. 6. Hyperparameter values history: (a) reglambda, and (b) R^2-score history of cross-validation.

Table 3
Execution time of algorithms.

Models	Training and testing (second)	Tuning (minutes)
GA-XGBoost	1.06	13.2
GA-CatBoost	0.48	8.6
GA-LightGBM	1.83	16.5

from the figure, the dispersion of data points around the mean value line is higher for walls that have a lower aspect ratio than 1.5. This is totally because of the more complicated behavior of these types of squat walls, which is known as shear-kind performance, and it is harder for ML models to learn these types of patterns [17]. However, as shown in the figure, the amount of dispersion is not excessive, indicating that the models performed well. The minimum, maximum, coefficient of variation, and standard deviation of

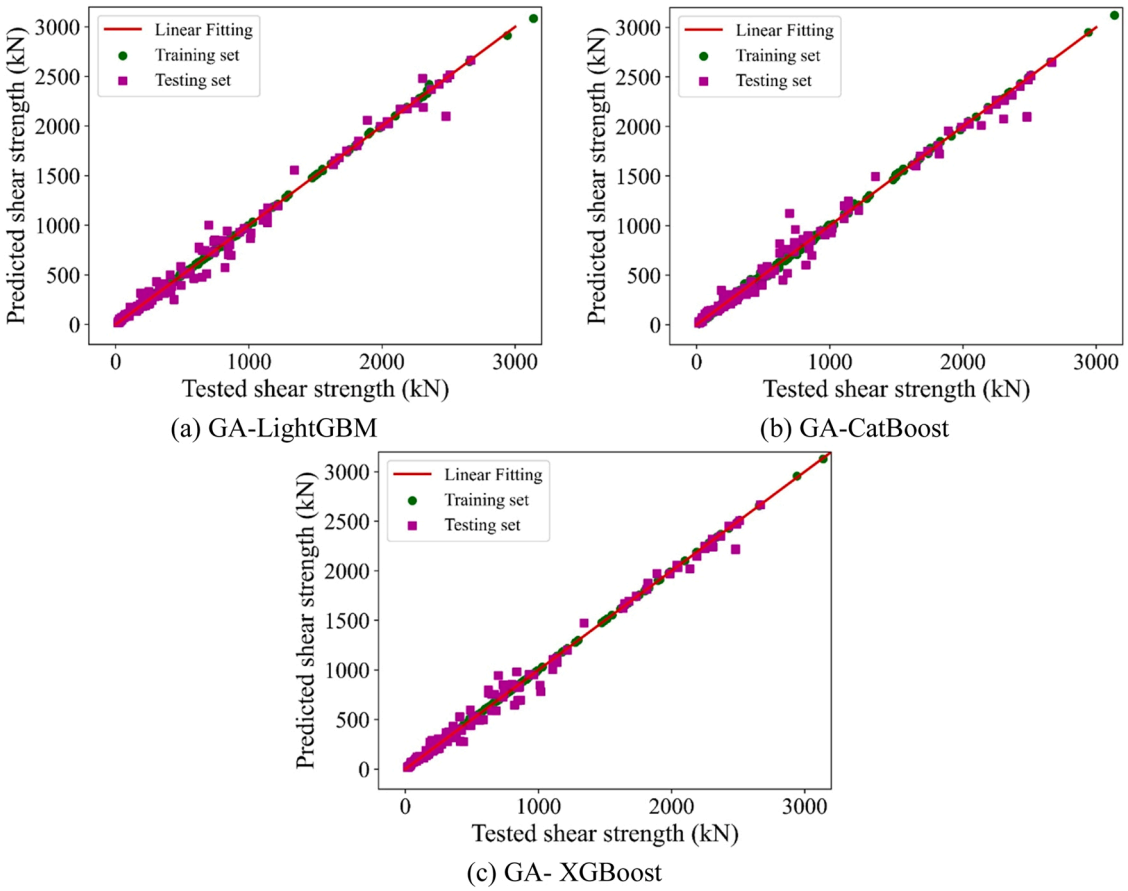

Fig. 7. Results of the shear strength prediction.

prediction-to-test ratios are also summarized in Table 4.

According to Table 4, the mean of prediction for all the three models is very close to 1.0. The highest coefficient of variation among all the three models is for XGBoost, which is only 8.1%, and for LightGBM and CatBoost are 10.6% and 13.6%, respectively. These results also show the excellent performance of GA tuned models, for shear strength prediction of shear walls.

3.2. Results on PCA extracted dataset

After showing that there are four pairs of features with a high correlation coefficient, for making a comparison between the number of necessary principal components for prediction, four and nine principal components with the cumulative explained variance equal to 0.72 and 0.95 are considered as new inputs for shear strength prediction and genetic-based ML models are examined using them. Singular value decomposition is used to obtain the relationship between PCA-selected features. Preprocessing stage is considered in the same manner as the original dataset, which means that 25% of the dataset is considered for the test set. Also, standard scaling is done for input features. Table 5. Displays the findings of five common metrics for algorithms based on the dataset obtained by PCA Analysis on the original dataset.

The preceding findings demonstrate the great results of genetic tuned boosting-based algorithms on PCA-based datasets, with at least 92.96% and 96.06% R^2 Scores for the GA-XGBoost and GA-CatBoost on four and nine principal component inputs, respectively. For four principal components GA-CatBoost and GA-LightGBM achieved the best results, while for nine components GA-XGBoost got the highest values from the measurement metrics, according to Table 5.

3.3. Validation of proposed model with other methods

3.3.1. Genetic programming (GP) based model

Gondia et al. [18], as mentioned before, used GP to develop a shear strength prediction model, by using a dataset of 254 squat reinforced concrete shear walls with boundary elements. In this study, important factors on the behavior walls and shear strength were identified in the first step. After that, GP, which is a form of artificial intelligence, was used to propose the expression. The final GP-generated expression, according to this study, is:

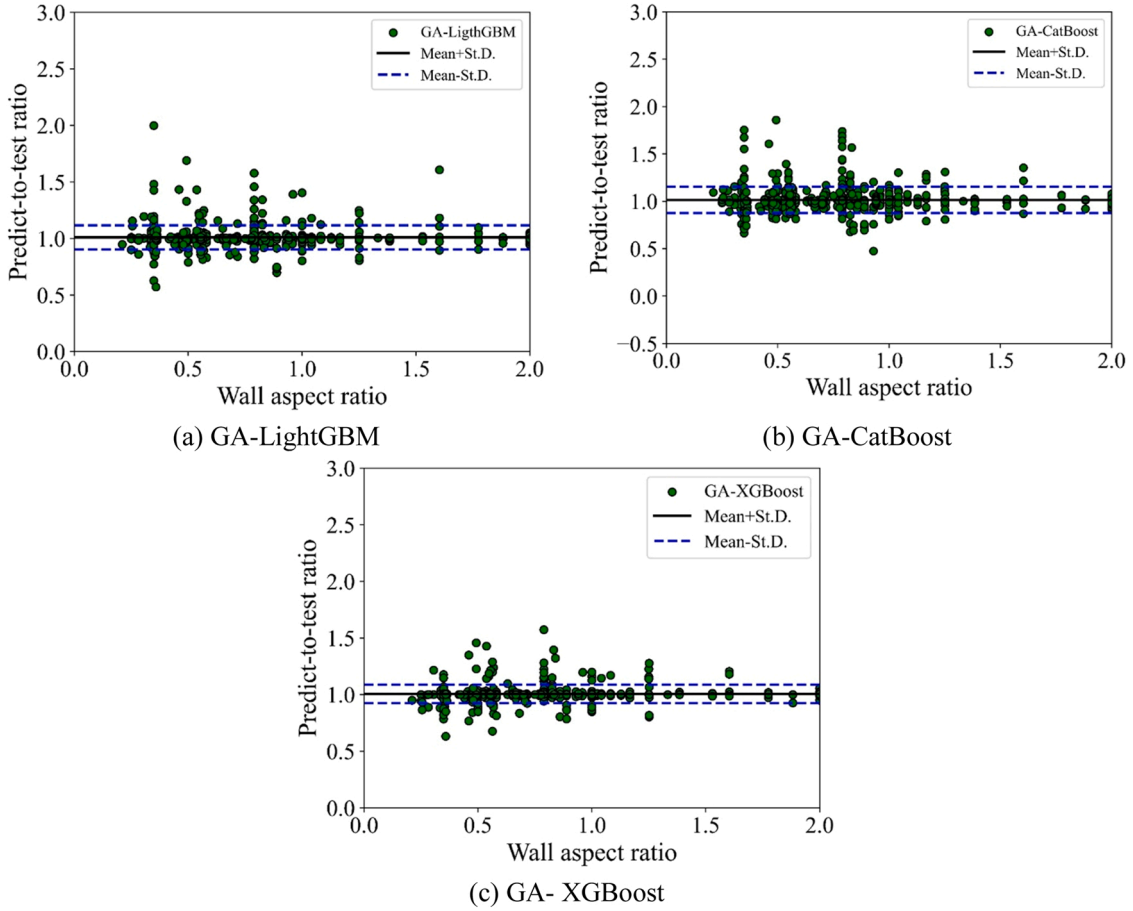

Fig. 8. Prediction –to– test ratios of shear strengths.

Table 4
Boosting-based methods of prediction performance.

Models	Predicted-to-test ratio results				
	Minimum	Maximum	Mean	Standard deviation	COV
GA-XGBoost	0.632	1.573	1.006	0.081	0.081
GA-CatBoost	0.474	1.858	1.014	0.138	0.136
GA-LightGBM	0.571	1.998	1.010	0.107	0.106

Table 5
Evaluation of different metrics on machine learning models with PCA dataset.

Number of Principal Components	Algorithms	Sets	R^2	RMSE (kN)	MAE (kN)	MAPE (%)	PI
Four	GA-XGBoost	Training	0.9984	28.49	16.76	8.59	0.015
		Testing	0.9296	178.55	113.23	40.53	0.109
	GA-CatBoost	Training	0.9959	14.45	10.83	6.41	0.008
		Testing	0.9538	144.74	81.72	25.16	0.085
	GA-LightGBM	Training	0.9845	81.88	47.58	19.54	0.049
		Testing	0.9450	136.92	76.88	34.76	0.093
Nine	GA-XGBoost	Training	0.9996	14.69	7.91	3.04	0.008
		Testing	0.9728	110.93	61.98	16.15	0.063
	GA-CatBoost	Training	0.9998	7.26	5.68	3.94	0.004
		Testing	0.9609	133.04	81.24	24.60	0.075
	GA-LightGBM	Training	0.9955	44.48	28.39	14.67	0.026
		Testing	0.9652	108.91	70.33	43.25	0.074

$$V_{GP} = \alpha_c\left(\sqrt{f'_c}\, t_w l_w\right) + \alpha_s\left(\rho_h f_{yh} t_w l_w\right) + \alpha_P(P) \tag{12}$$

where:

$$\alpha_c = [8.29 - 1.14(h_w/l_w)]$$

$$\alpha_s = [0.68 - 0.38(h_w/l_w)]$$

$$\alpha_P = [0.44 - 0.17(h_w/l_w)]$$

The above expression is valid for walls with the same values of ρv and ρh, according to the mentioned study; thus, a subset of the original dataset with this condition is extracted to do a comparative study with current methods. The range of variables in the subset is also set to be nearly the same as the training and testing dataset of GP expression. The statistical information of this part is summarized in Table 6.

First, the GP-based expression is used to predict the shear strength of wall samples in Table 6. After that, the results of genetic tuned algorithms with this subset are extracted. Compression is done based on the mean, standard deviation, and coefficient of variation of the predicted to test values ratio. The results are summarized in Table 7.

As seen from the previous table, the performance of the three GA tuned boosting-based methods is better than GP-based expression, with relative mean values close to 1.0 and COVs of 0.11, 0.10, and 0.16, for GA-LightGBM, GA-XGBoost, and GA-CatBoost, respectively. It must be noted that the resulted mean and COV of GP expression are close to what is mentioned in the reference paper, which is 0.27 and 0.26, for training and testing sets, respectively.

3.3.2. Mechanic-based semi-empirical model

A mechanic-based semi-empirical model, which is based on the ASCE/SEI 43–05 [53], is adopted to examine the accuracy of ML models. The formulation of the mentioned model is as (Eq. 13):

$$V_n = v_n d t_w$$

$$v_n = 0.69\sqrt{f'_c} - 0.28\sqrt{f'_c}\left(\frac{h_w}{l_w} - 0.5\right) + \frac{P}{4 l_w t_w} + \rho_{se} f_{yh} \le 1.66\sqrt{f'_c}\rho_{se} = A\rho_v + B\rho_h \tag{13}$$

In the above equations, $d = 0.6 l_w$; ρ_{se} is the equivalent reinforcing ratio that combines ρ_h and ρ_v. A and B in the equation of ρ_{se}, can be obtained as follow:

$$\left\{\begin{array}{c} h_w/l_w \le 0.5, A = 1, B = 0 \\ h_w/l_w \ge 1.5, A = 0, B = 1 \\ 0.5 \le h_w/l_w \le 1.5, A = -h_w/l_w + 1.5, h_w/l_w - 0.5 \end{array}\right\}$$

This model, combined with GA-LightGBM, GA-CatBoost and GA-XGBoost, is used to predict the shear strength of 558 shear walls in the dataset.

The mean, standard deviation, maximum, minimum, and COV values for these predicted-to-test ratios are shown in Table 8**.** The results of the three ML models offer a superior mean prediction, and a considerably lower prediction variance, as seen in the table. The ASCE model's mean ratio prediction is 0.87, which is close to 1.0; however, the model's coefficient of variation is relatively large (0.43). The results from ML models are excellent for both mean and coefficient of variation, which shows the robustness of the proposed models.

3.4. Sensitivity and parametric analysis

A parametric analysis based on Gandomi et al. [37] was also performed for each of the three boosting-based methods, giving the possibility to investigate the influences of each input variable on the final shear strength estimation output of the models. The sensitivity of model output for prediction to each of the input variables can be investigated using E.q.14 and E.q.15:

$$N_i = f_{\max}(x_i) - f_{\min}(x_i) \tag{14}$$

$$S_i = \frac{N_i}{\sum_{j=1}^{n} N_j} \times 100 \tag{15}$$

where $f_{\max}(x_i)$and $f_{\min}(x_i)$ are, respectively, the maximum and minimum wall shear strength estimations resulted based on the predictive models while setting all other variables except x_i equal to their mean values.

Results of the parametric study are shown in Fig. 9. As seen in this figure, all three boosting-based methods show obvious positive trends for features, like (Lw), (tw), (P) and (bf). These are parameters that when increasing their value, affect positively the shear strength of walls and increase shear strength estimation. Looking to feature like (Hw), there is decreasing trends for shear strength estimation as its value will decrease with increasing values of this feature. Also, slight decreasing trends can be deduced for some ranges of (fyh). However, as can be seen, trends of shear strength prediction for each input feature are not smooth due to the tree-based

Table 6
Wall samples in the reduced dataset.

Parameters/value	count	mean	std	min	25%	50%	75%	max
Hw	187	753.694	255.419	330.000	488.975	800	1000	1200
Lw	187	1384.968	685.904	430	813	1220	1905.500	3327
tw	187	62.909	35.460	10	40	50.800	79	160
fc	187	29.520	14.120	12.300	20.690	25.580	34.995	102
ρv	187	0.008	0.005	0	0.005	0.006	0.012	0.024
fyv	187	366.574	74.011	270.973	323	341.302	412	551.600
ρh	187	0.008	0.005	0	0.005	0.006	0.012	0.024
fyh	187	365.546	71.266	271	323	341.300	412	500
ρL	187	0.034	0.017	0.006	0.021	0.032	0.047	0.089
fyL	187	364.474	68.774	272	308.850	366.800	407	539.200
P	187	176.014	359.811	0	0	0	30.020	1423
tf	187	113.040	63.633	30	60	101.600	127	320
bf	187	234.550	224.013	60	95.250	145	300	1000

Table 7
Compression of three ML models with GP-Based expression.

Models	Predicted-to-test ratio results				
	Minimum	Maximum	Mean	Standard deviation	COV
GP-Based expression	0.57	2.65	1.03	0.31	0.30
GA-XGBoost	0.676	1.573	1.016	0.104	0.102
GA-CatBoost	0.475	1.858	1.029	0.165	0.161
GA-LightGBM	0.815	1.689	1.013	0.108	0.107

Table 8
Compression of the three ML models with a semi-empirical model.

Models	Predicted-to-test ratio results				
	Minimum	Maximum	Mean	Standard deviation	COV
ASCE	0.31	2.91	0.87	0.38	0.43
GA-XGBoost	0.632	1.573	1.006	0.081	0.081
GA-CatBoost	0.474	1.858	1.014	0.138	0.136
GA-LightGBM	0.571	1.998	1.010	0.107	0.106

structure of these algorithms. Also, it must be noted that models converge to a constant behavior for low-frequency ranges of feature values, and cannot have a good training. In general, the performance of boosting-based methods, like other complex ML models are so dependent on the availability of a sufficient amount of data for both training and testing so that it will ensure both high performance and no occurrence of overfitting and also proper tuning of these algorithms is necessary to avoid overfitting. The results of the sensitivity analysis of the three models are also depicted in Fig. 10, which shows that the shear strength prediction results of GA-LightGBM and GA-CatBoost are most sensitive to (Lw), (tw), (bf) and (P). For GA-XGBoost, the four most sensitive parameters are (Lw), (tw), (ρh) and (Hw). These findings are in close agreement with other studies like [17] and [18], which show the robustness of the three models.

4. Conclusions

This study introduced a boosting-based technique, to estimate the shear strength of squat RC walls, which was tuned using a GA. For this experiment, a dataset of 558 specimens were used; and then, the dataset was randomly split into training and test sets, and preprocessing methods were applied. Then, an ML-tuned algorithm was used to predict shear strength, and the results were evaluated with other methods using five accuracy measurements. It was also studied the presence of multicollinearity in the dataset, having found four pairs of features with high correlation between them. PCA analysis was performed to tackle multicollinearity, four and nine principal components were considered as new input features. Also, the prediction results of ML models on these new reduced datasets were discussed. The proposed models' robustness was examined using another GP and semi-empirical model for shear strength prediction. Finally, sensitivity and parametric analyses for the three studied models were conducted.

The following are the conclusions that could be derived from this research:

- Due to the excellent accuracy and speed that can be achieved, boosting-based Ensemble learning algorithms, like LightGBM, XGBoost, and CatBoost, are beneficial in prediction problems. The validation accuracy of these methods was as high as 99%.

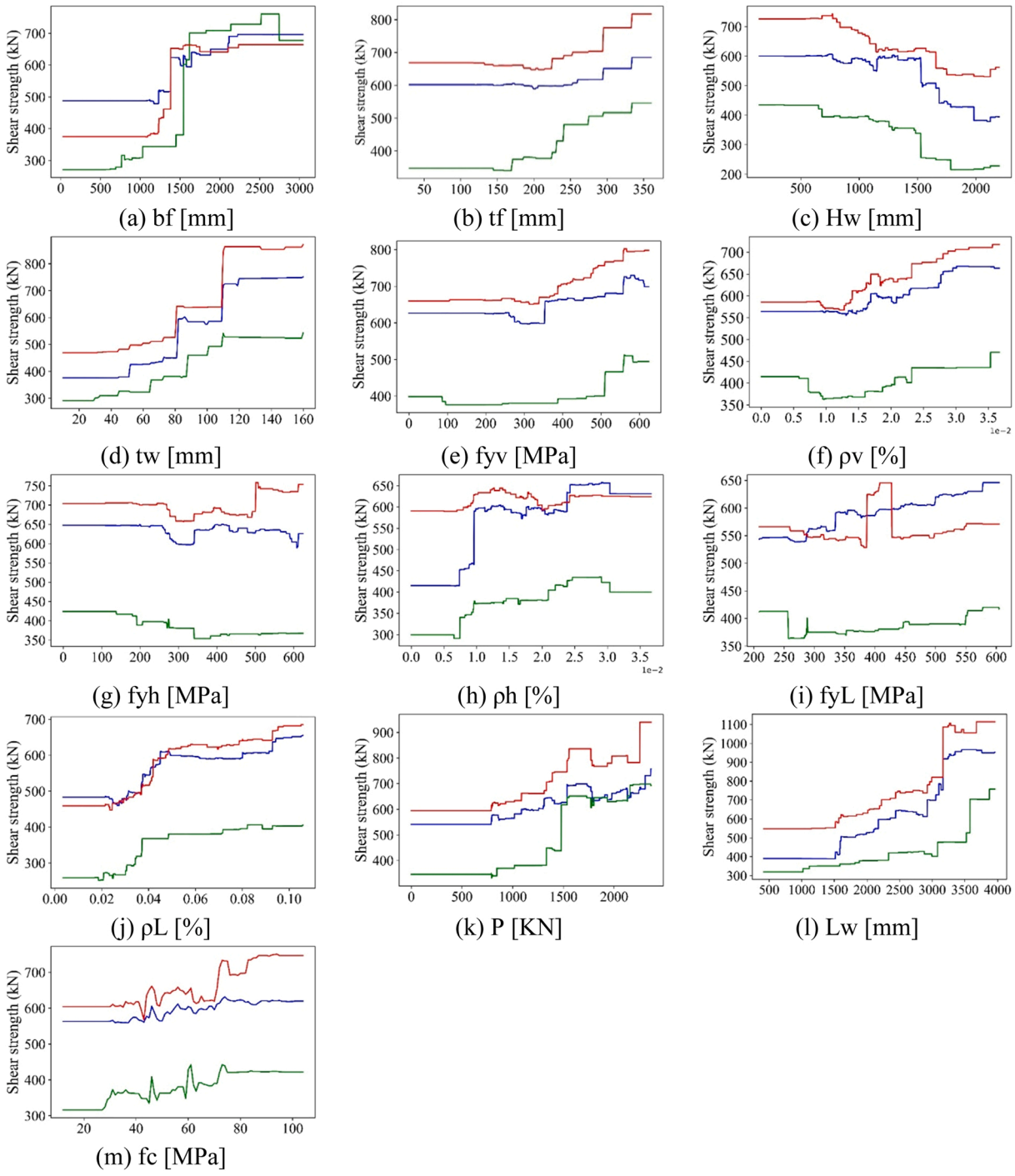

Fig. 9. Parametric analysis of dataset shear strength using developed GA-XGBoost (blue line), GA-CatBoost (green line), GA- LightGBM (red line).

- For hyperparameter tuning of models with a high search space, a metaheuristic approach like a GA can be more efficient, when compared with other methods like grid search, due to speed and automatic capabilities.
- A compression is made between the Genetic tuned LightGBM, XGBoost, and CatBoost method; XGBoost achieved the best overall performance with $R^2 = 0.9906$, RMSE = 71.38 kN, MAE = 43.89 kN, MAPE = 11.44 %, and PI = 0.036 on the testing set.
- The PCA method can be useful on datasets with a high correlation between features. The predictions made using these new features were made with advanced ML methods, leading to the conclusion that extracted datasets can be close to full-dimensional ones. This is very important for datasets with a high number of features. Four and nine principal components of the RC squat wall dataset are

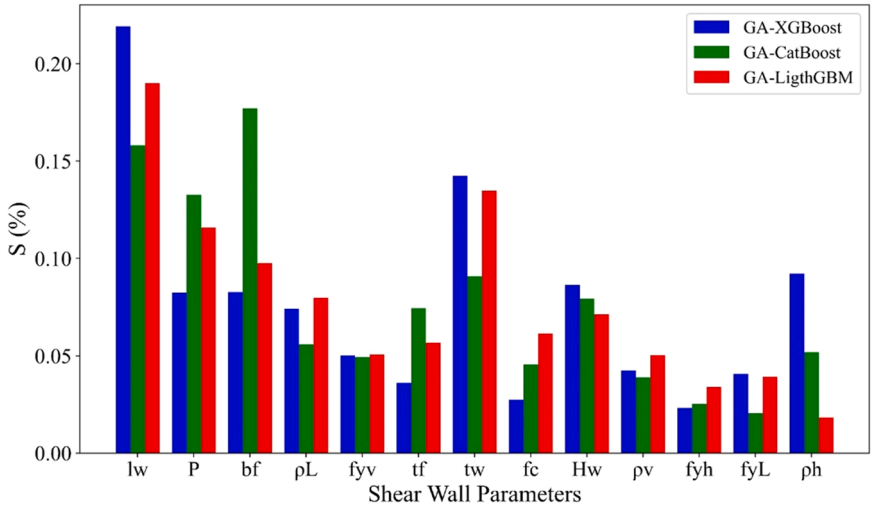

Fig. 10. Sensitivity analysis of input features for GA-XGBoost, GA-CatBoost, and GA-LightGBM.

considered as new input features, and three models are investigated on them. Genetic-tuned CatBoost had the best results for four principal components, while GA-XGBoost scored the best for a dataset with nine considered principal components.

- Compression is performed between three proposed models and GP-based expression based on a reduced dataset compatible with feature ranges used for training and testing GP-based expression. ML models have much less coefficient of variation and mean values close to one for predicted to test ratios of shear strength. These results are the same for a semi-empirical model based on ASCE.
- Sensitivity and parametric analysis were also conducted for three models. The results show that models are sensitive to critical parameters on the shear strength of squat walls, which is comparable with other studies in this area.

Due to the excellent accuracy and speed that can be achieved, boosting-based ensemble learning algorithms might be beneficial as surrogate models in prediction and optimization problems. Future use of these algorithms in studies like reliability-based design optimization can greatly reduce the required computational cost. Also, they can eliminate experimental investigations of squat shear walls with configurations in the range of the training dataset. However, to produce the same high accuracy for models on other ranges of data and other types of shear walls, additional studies are required based on new sample data points.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] B. Li, Z. Pan, W. Xiang, Experimental evaluation of seismic performance of squat RC structural walls with limited ductility reinforcing details, J. Earthq. Eng. 19 (2) (2015) 313–331.
[2] B. Li, K. Qian, H. Wu, Flange effects on seismic performance of reinforced concrete squat walls with irregular or regular openings, Eng. Struct. 110 (2016) 127–144.
[3] W. Kassem, Shear strength of squat walls: A strut-and-tie model and closed-form design formula, Eng Struct 84 (2015) 430–438.
[4] C. Ning, B. Li, Probabilistic development of shear strength model for reinforced concrete squat walls, Earthq. Eng. Struct. Dyn. 46 (6) (2017) 877–897.
[5] H.-W. Yu, S.-J. Hwang, Evaluation of softened truss model for strength prediction of reinforced concrete squat walls, J. Eng. Mech. 131 (8) (2005) 839–846.
[6] L.M. Massone, Strength prediction of squat structural walls via calibration of a shear–flexure interaction model, Eng. Struct. 32 (4) (2010) 922–932.
[7] C.K. Gulec, A.S. Whittaker, B. Stojadinovic, Shear strength of squat rectangular reinforced concrete walls, Acids Struct. J. 105 (4) (2008) 488.
[8] W.W. El-Dakhakhni, B.R. Banting, S.C. Miller, Seismic performance parameter quantification of shear-critical reinforced concrete masonry squat walls, J. Struct. Eng. 139 (6) (2013) 957–973.
[9] H. Sun, H.V. Burton, H. Huang, Machine learning applications for building structural design and performance assessment: State-of-the-art review, Journal of Building Engineering 33 (2021) 101816.
[10] S. Mangalathu, J.-S. Jeon, Machine learning–based failure mode recognition of circular reinforced concrete bridge columns: Comparative study, Journal of Structural Engineering 145 (10) (2019) 4019104.
[11] S. Mangalathu, S.-H. Hwang, J.-S. Jeon, Data-driven machine-learning-based seismic failure mode identification of reinforced concrete shear walls, Eng Struct 208 (2020), 110331.

[12] A. Siam, M. Ezzeldin, W. El-Dakhakhni, Machine learning algorithms for structural performance classifications and predictions: Application to reinforced masonry shear walls, Structures 22 (2019) 252–265.
[13] D.-C. Feng, Z.-T. Liu, X.-D. Wang, Z.-M. Jiang, S.-X. Liang, Failure mode classification and bearing capacity prediction for reinforced concrete columns based on ensemble machine learning algorithm, Advanced Engineering Informatics 45 (2020) 101126.
[14] D.-C. Feng, B. Cetiner, M.R. Azadi Kakavand, E. Taciroglu, Data-driven approach to predict the plastic hinge length of reinforced concrete columns and its application, Journal of Structural Engineering 147 (2) (2021) 4020332.
[15] T.-T. Le, M.V. Le, Development of user-friendly kernel-based Gaussian process regression model for prediction of load-bearing capacity of square concrete-filled steel tubular members, Mater Struct 54 (2) (2021) 1–24.
[16] Y. Wu, Y. Zhou, Hybrid machine learning model and Shapley additive explanations for compressive strength of sustainable concrete, Constr Build Mater 330 (2022) 127298.
[17] D.-C. Feng, W.-J. Wang, S. Mangalathu, E. Taciroglu, Interpretable XGBoost-SHAP Machine-Learning Model for Shear Strength Prediction of Squat RC Walls, Journal of Structural Engineering 147 (11) (2021) 4021173, https://doi.org/10.1061/(asce)st.1943-541x.0003115.
[18] A. Gondia, M. Ezzeldin, W. El-Dakhakhni, Mechanics-guided genetic programming expression for shear-strength prediction of squat reinforced concrete walls with boundary elements, Journal of Structural Engineering 146 (11) (2020) 4020223.
[19] M. Sarveghadi, A.H. Gandomi, H. Bolandi, A.H. Alavi, Development of prediction models for shear strength of SFRCB using a machine learning approach, Neural Comput Appl 31 (7) (2019) 2085–2094.
[20] N. Aravind, S. Nagajothi, S. Elavenil, Machine learning model for predicting the crack detection and pattern recognition of geopolymer concrete beams, Constr Build Mater 297 (2021) 123785.
[21] R. Elshawi, M. Maher, and S. Sakr, "Automated machine learning: State-of-the-art and open challenges," arXiv preprint arXiv:1906.02287, 2019.
[22] M. Kuhn, K. Johnson, Applied predictive modeling, 26, Springer, 2013.
[23] G.I. Diaz, A. Fokoue-Nkoutche, G. Nannicini, H. Samulowitz, An effective algorithm for hyperparameter optimization of neural networks, IBM J Res Dev 61 (4/5) (2017) 1–9.
[24] F. Hutter, L. Kotthoff, J. Vanschoren, Automated machine learning: methods, systems, challenges, Springer Nature (2019).
[25] S. Abreu, "Automated architecture design for deep neural networks," arXiv preprint arXiv:1908.10714, 2019.
[26] O.S. Steinholtz. A comparative study of black-box optimization algorithms for tuning of hyper-parameters in deep neural networks, Luleå University of Technology, 2018.
[27] Q. Yao et al., "Taking human out of learning applications: A survey on automated machine learning," arXiv preprint arXiv:1810.13306, 2018.
[28] S. Lessmann, R. Stahlbock, S.F. Crone. Optimizing hyperparameters of support vector machines by genetic algorithms, IC-AI, 2005, pp. 74–82.
[29] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in Proceedings of the genetic and evolutionary computation conference, 2017, pp. 481–488.
[30] L. Yang, A. Shami, On hyperparameter optimization of machine learning algorithms: Theory and practice, Neurocomputing 415 (2020) 295–316.
[31] M. Jain, "Hyperparameter tuning in XGBoost using genetic algorithm", [Online]. Available: https://towardsdatascience.com/hyperparameter-tuning-in-xgboost-using-genetic-algorithm-17bd2e581b17.
[32] A. Telikani, A. Tahmassebi, W. Banzhaf, A.H. Gandomi, Evolutionary Machine Learning: A Survey, ACM Computing Surveys (CSUR) 54 (8) (2021) 1–35.
[33] J. Zhang, Y. Sun, G. Li, Y. Wang, J. Sun, J. Li, Machine-learning-assisted shear strength prediction of reinforced concrete beams with and without stirrups, Eng Comput (2020) 1–15.
[34] Y. Sun, et al., Determination of Young's modulus of jet grouted coalcretes using an intelligent model, Eng Geol 252 (2019) 43–53.
[35] Y. Sun, G. Li, N. Zhang, Q. Chang, J. Xu, J. Zhang, Development of ensemble learning models to evaluate the strength of coal-grout materials, Int J Min Sci Technol 31 (2) (2021) 153–162.
[36] A.H. Gandomi, G.J. Yun, A.H. Alavi, An evolutionary approach for modeling of shear strength of RC deep beams, Mater Struct 46 (12) (2013) 2109–2119.
[37] M.A. Jia-xing, C.H.E.N. Ke-yu, W.A.N.G. Yin-hui, L.I. Bing, Peak shear strength of H-shaped reinforced concrete squat walls, 工程力学 38 (4) (2021) 123–135. Chicago.
[38] J. Ma, C.-L. Ning, B. Li, Peak shear strength of flanged reinforced concrete squat walls, Journal of Structural Engineering 146 (4) (2020) 04020037.
[39] A. Gogna, A. Tayal, Metaheuristics: review and application, J. Exp. Theor. Artif. Intell. 25 (4) (2013) 503–526.
[40] K. Eggensperger, et al., Towards an empirical foundation for assessing bayesian optimization of hyperparameters, NIPS Workshop Bayesian Optim. Theory Pract. 10 (3) (2013).
[41] A.R. Kashani, C.V. Camp, M. Rostamian, K. Azizi, A.H. Gandomi, Population-based optimization in structural engineering: a review, Artif. Intell. Rev. (2021) 1–108.
[42] S. Li, F. Jiang, Y. Qin, K. Zheng, Social spammer detection based on PSO-CatBoost, International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage (2020) 382–395.
[43] F. Itano, M.A. de A. de Sousa, E. Del-Moral-Hernandez, Extending MLP ANN hyper-parameters optimization by using genetic algorithm, 2018 International Joint Conference on Neural Networks (IJCNN) (2018) 1–8.
[44] B. Kazimipour, X. Li, A.K. Qin, A review of population initialization techniques for evolutionary algorithms, 2014 IEEE Congress on Evolutionary Computation (CEC) (2014) 2585–2592.
[45] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, A novel population initialization method for accelerating evolutionary algorithms, Comput. Math. Appl. 53 (10) (2007) 1605–1614.
[46] F.G. Lobo, D.E. Goldberg, M. Pelikan, Time complexity of genetic algorithms on exponentially scaled problems, Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation (2000) 151–158.
[47] A. Tahmassebi, M. Motamedi, A.H. Alavi, and A.H. Gandomi, An explainable prediction framework for engineering problems: case studies in reinforced concrete members modeling, Eng. Comput., 2021.
[48] T. Chen, C. Guestrin, XGBoost: a scalable tree boosting system, Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining (2016) 785–794.
[49] G. Ke, et al., Lightgbm: a highly efficient gradient boosting decision tree, Adv. Neural Inf. Process. Syst. 30 (2017).
[50] Y. Wang, T. Wang, Application of improved LightGBM model in blood glucose prediction, Appl. Sci. 10 (9) (2020) 3227.
[51] H. Abdi, L.J. Williams, Principal component analysis, Wiley Interdiscip. Rev. Comput. Stat. 2 (4) (2010) 433–459.
[52] A.H. Gandomi, D.A. Roke, Assessment of artificial neural network and genetic programming as predictive tools, Adv. Eng. Softw. 88 (2015) 63–72.
[53] N. S. Committee, Seismic Design Criteria for Structures, Systems, and Components in Nuclear Facilities, Am. Soc. Civ. Eng. Reston, VA, 2005.