Efficient machine learning models for prediction of concrete strengths

Hoang Nguyen[a,*], Thanh Vu[b], Thuc P. Vo[c,*], Huu-Tai Thai[d]

[a]*Department of Mechanical and Construction Engineering, Northumbria University, Newcastle upon Tyne NE1 8ST, United Kingdom*
[b]*Australian e-Health Research Centre, CSIRO, Brisbane, Australia*
[c]*School of Engineering and Mathematical Sciences, La Trobe University, Bundoora, VIC 3086, Australia*
[d]*Department of Infrastructure Engineering, The University of Melbourne, Parkville VIC 3010, Australia*

Abstract

In this study, an efficient implementation of machine learning models to predict compressive and tensile strengths of high-performance concrete (HPC) is presented. Four predictive algorithms including support vector regression (SVR), multilayer perceptron (MLP), gradient boosting regressor (GBR), and extreme gradient boosting (XGBoost) are employed. The process of hyperparameter tuning is based on random search that results in trained models with better predictive performances. In addition, the missing data is handled by filling with the mean of the available data which allows more information to be used in the training process. The results on two popular datasets of compressive and tensile strengths of high performance concrete show significant improvement of the current approach in terms of both prediction accuracy and computational effort. The comparative studies reveal that, for this particular prediction problem, the trained models based on GBR and XGBoost perform better than those of SVR and MLP.

Keywords: High performance concrete, Ensemble learning, Support vector machine, Multi-layer Perceptron, Tree-based algorithms

*Corresponding author

Email addresses: hoang.nguyen@northumbria.ac.uk (Hoang Nguyen), t.vo@latrobe.edu.au (Thuc P. Vo)

1. Introduction

Concrete has been widely used in building and civil structures as it possesses many desired engineering properties. The high strength when combined with reinforcement and the ability to cast into shapes as well as harden at ambient temperature enable concrete to be a prominent choice in constructing structural elements of apartments and high-rise buildings. In addition, high temperature and excellent water resistance are also cited as the advantages of concrete which allow reinforced concretes to be the materials of choice for structures that regularly expose to extreme environmental impacts such as tunnel, bridges, dams, reservoirs, and the like. Another reason making concrete a popular material in construction lies in its economic aspects. Regular concrete is basically made of coarse aggregate, e.g. rock, fine aggregate, e.g. sand, binding material, e.g. cement, and water. All of which are not expensive and can be found locally in the area of construction. This shows remarkable advantages compared to other building materials such as steel where structural elements required to be processed in well-equipped factories with the involvement of different machines. Furthermore, in order to make concrete a better material with higher engineering performance, fly ash, blast furnace slag and other supplementary substances are added to the mixture [1, 2]. The addition of these industrial wastes can considerably reduce the environmental impact without compromises in structures' integrity which in turn increases the sustainability of concrete.

One of the issues of concrete material, in general, is the content selection and the prediction of its output engineering properties including compressive and tensile strengths. This is because concrete, especially high-performance concrete, is a highly nonhomogeneous mixture with different constituents. Therefore, it is vital to have robust and reliable predictive models based on existing input and output data at the early stage to drive down the cost of making further experiments. Appropriate predictive models also allow reductions of trivial attempts in searching for appropriate input combinations that can potentially lead to desirable concrete performances. Consequently, they enable significant time and cost savings. Due to the highly nonlinear relation between the input constituents and the output of concrete strengths, creating such models is a challenging task.

There have been significant efforts to utilise smart computing algorithms to tackle civil engineering problems in the last few decades as suggested in the brief review of Rafiei [3]. Data-driven approaches have been used to analyse structural behaviours [4, 5].

In estimating material properties, researchers have established predictive models with the ultimate goal is to minimise the prediction error against the actual data collected from experiments. Ni and Wang proposed a multilayer feedforward neural networks to predict the compressive strength of concrete [6]. The method was utilised to deal with the nonlinear relationship between the input features and the concrete strength. Rafiei et al. used a nonlinear optimisation algorithm and a computational intelligence-based classification algorithm to solve the concrete mixture design problem in which desired constraints were taken into account [7]. The same authors also proposed statistical and neural network models to estimate concrete properties based on input parameters [8]. Yeh and Lien presented a knowledge discovery method namely Genetic Operation Tree as a combination of the operation tree and genetic algorithm to estimate concrete's compressive strength via self-organised formulas [9]. In this model, while the operation tree is used to build an explicit formula, the genetic algorithm is employed to search for optimal parameters used in the operation tree. There are also different approaches that can be used to predict strengths of HPC which include data-mining techniques [10], enhanced artificial intelligence for ensemble approach [11], metaheuristic regression system [12, 13]. Engen et al. [14] employed a hierarchical model to predict the variability of material properties in ready-mixed concrete. Erdel et al. incorporated bagging and gradient boosting techniques to construct ensemble models based on the discrete wavelet transform [15]. This enhanced combination was then used to forecast the compressive strength of HPC. There are also recent developments on using network-related and optimisation algorithms to predict material properties [16–18]. Recently, Bui et al. employed artificial neural network (*ANN*), in which firefly algorithm was used to search for optimal network parameters, to predict both compressive and tensile strengths of HPC [19]. Nguyen et al. proposed a high-order artificial neural network, in which high-order neuron was employed, to predict foamed concrete strengths including the compressive strength of HPC [20].

The present study focuses on proposing a highly efficient implementation of machine learning model that enables the achievement of optimal *hyperparameters*, which are initialised at the beginning and kept unchanged during the training process of the machine learning algorithms in a large search space. It should be noted that the hyperparameter initialisation plays an important role to the success of the machine learning models [21, 22]. In addition, a proposed method of handling missing data using single mean imputation significantly improves the predictive results. These two main contributions are briefly highlighted as follows.

Firstly, this study presents efficient implementation and evaluates the performance of support vector regression (*SVR*) [23–25], multilayer perceptron (*MLP*) [26, 27], gradient boosting regressor (*GBR*) [28], and extreme gradient boosting (*XGBoost*) [29] which give considerable improvement in terms of both prediction accuracy and computational efficiency. The performances of the prediction models for HPC compressive and tensile strengths are considerably improved in comparison with those existing in the literature. This is due to the efficient implementation in which open-sourced machine learning libraries of *scikit-learn* [30] and *XGBoost* [29] are involved. This combination allows significantly less computing effort enabling more spaces and resources for hyperparameter tuning. The comparison studies between the performance of the four machine learning techniques reveals the advantages of GBR and XGBoost over SVR and MLP both in terms of accuracy and efficiency.

Secondly, a proposed method for handling missing data by using the mean of the available data significantly increases predictive performance. Experimental results on the HPC tensile strength dataset, in which missing data ranges from 0% to 39% show that the proposed method achieves the new state-of-the-art RMSE that is considerably lower than the current best reported in the literature.

The outline of the remaining of this study is as follows. Section 2 gives a brief review on SVR, MLP, GBR, and XGBoost while Section 3 discusses the assessment of the datasets of HPC. Section 4 presents the implementation and results of the predictive models as well as discussion on how hyperparameters affect their performance. The study is closed with concluding remarks which are given in Section 5.

2. Review on machine learning algorithms/techniques

This study aims to show the importance of *hyperparameter* initialisation and handling *missing data*. Four popular machine learning algorithms in data mining and civil engineering, therefore, are employed to handle the HPC problems. Specifically, they include (i) Support vector regression (SVR) which is support vector machine (SVM) used in regression applications, (ii) Multilayer perceptron (MLP) which is also known as a deep feedforward network is essentially a typical deep artificial neural network, (iii) Gradient boosting machines (GBMs) or gradient tree boosting including Gradient Boosting Regressor (GBR) and Extreme Gradient Boosting (XGBoost) which are well-known tree-based ensemble models. Fig. 1 shows the general architecture of the machine learning models. For example, in the HPC compressive strength prediction task, the features consist of Cement, Blast furnace slag, Fly ash, Water, Superplasticizer, Coarse aggregate, Fine aggregate, Age and Compressive strength. The output is a predicted real number of the compressive strength produced by the machine learning model regarding the input features.

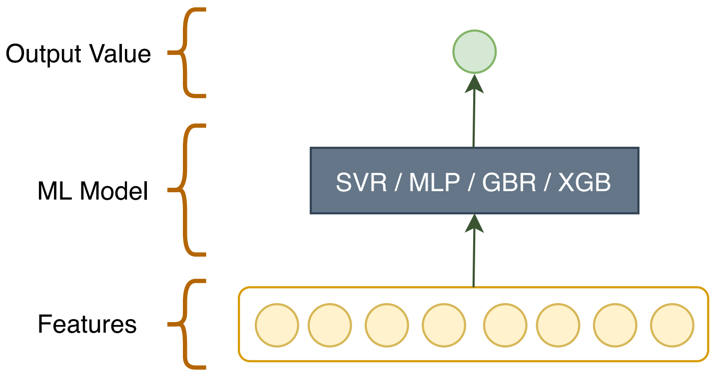


Figure 1: The machine learning model architecture for the presenting HPC regression problems.

2.1. Support vector machine

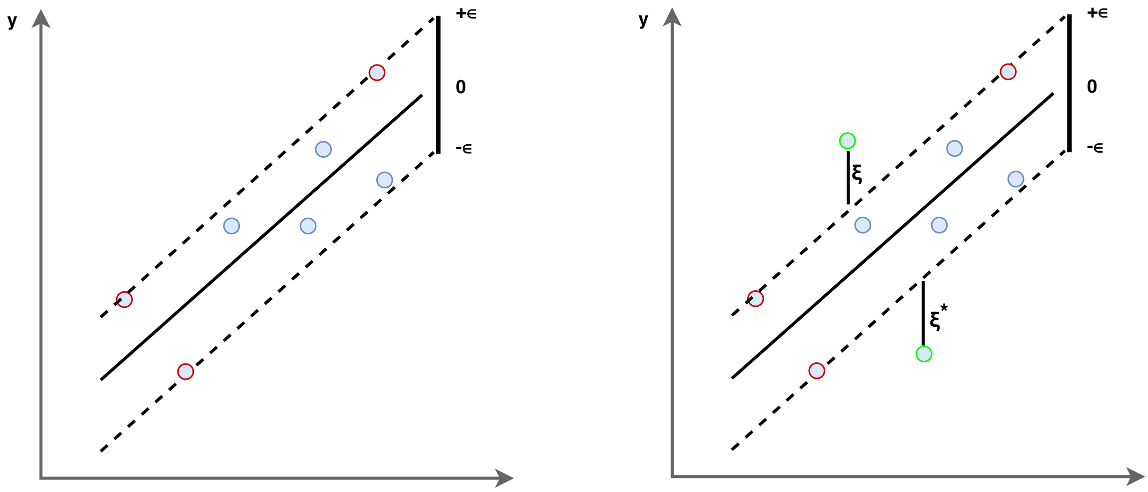
SVM is a well-known supervised machine learning model which was developed largely by Vapnik and his colleagues at AT&T Bell Laboratories in the 1990s [23, 24, 31]. The key idea of SVM is that it maps the input vectors into a high dimensional feature space using some nonlinear *kernel function*, chosen a *prior*, so called a *hyperparameter* which is the parameter initialised before and fixed during the training of the machine learning model. In this feature space, a linear decision surface is constructed with a property of ensuring high generalisation of the learning machine [24]. SVM has been widely used and obtained high performances in both classification and regression applications [25]. When SVM is utilised in regression applications, it is called *support vector regression (SVR)* [25].

In this study, ϵ-SVR proposed by [23] is utilised to handle the HPC regression problems. Specifically, given training data $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \subset \chi \times \mathbb{R}$, where χ denotes the space of input features (e.g., $\chi = \mathbb{R}^d$, here d is the number of input features), the goal of ϵ-SVR is to find the function $f(x_i)$ that has at most ϵ deviation from the actual target value y_i for all n samples in the training data [25]. Assuming that f is a linear function of the form

$$f(x) = \langle w, x \rangle + b, \quad w \in \mathbb{R}, b \in \mathbb{R}, \tag{1}$$

where $\langle ., . \rangle$ denotes the dot function. A small w is sought to make the function f flat by minimising the norm, i.e. $\|w\|^2 = \langle w, w \rangle$ as follows

$$\begin{aligned} &\text{minimise} \quad \frac{1}{2}\|w\|^2, \\ &\text{subject to} \quad |y_i - \langle w, x_i \rangle - b| \leq \epsilon. \end{aligned} \tag{2}$$



(a) Without "soft margin" (b) With "soft margin"

Figure 2: Examples of solvable (a) and non-solvable (b) problems by standard SVM in 2D space. Red circles are support vectors, green circles denote data points that do not satisfy Eq. (2).

Fig. 2a shows an example of a solvable problem in 2D space in which all input pairs (x_i, y_i) satisfy Eq. (2). However, solving this equation that satisfies for all pairs (x_i, y_i) is not always feasible [24, 25] due to the large amount of data of a practical problem and some data points are just out of the supported range bounded by ϵ as shown in Fig. 2b. Some errors in the model should be allowed which leads to the idea of using "soft margin" in SVM proposed by Cortes and Vapnik [24]. This is done by introducing slack variables ξ_i, ξ_i^* to handle the infeasible constraints. Hence, SVR can be formulated as follows

$$\begin{aligned} &\text{minimise} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*), \\ &\text{subject to} \quad \begin{cases} |y_i - \langle w, x_i \rangle - b| \leq \epsilon, \\ \xi_i, \xi_i^* \geq 0. \end{cases} \end{aligned} \tag{3}$$

The constant $C > 0$ determines the trade-off between the *flatness* of the function f and the amount up to which deviations larger than ϵ are tolerated [25]. Fig. 2b shows an example of using "soft-margin" to handle a regression problem in a 2D space.

Although using "soft margin" allows some errors when training the model with the linear form of f, this function is not always available [23]. To tackle this issue, one can make the SVR algorithm nonlinear and this could be done by utilising a *kernel function* to transform the original data from a low dimensional vector space to a higher dimensional vector space where a linear form of f can be found. The kernel function, a hyperparameter, can be *linear*, *polynomial*, *radial basic - rbf* or *sigmoid* function [25]. Interested readers are referred to the "Tutorial on Support Vector Regression" [25] for more information on kernel functions.

It is noted that some extensions of SVR have been proposed and obtained high performances in the HPC regression applications [11, 12, 32]. in which the authors focused on modifying the model architecture. In this current study, tuning *hyperparameters* including `epsilon` - ϵ, the "soft margin" constant - `C`, the kernel function - `kernel`, and the kernel coefficient parameter `gamma` - γ, which are largely ignored in the previous research [21, 22], are investigated.

2.2. Multilayer perceptron

MLP or a deep feedforward network is a typically deep ANN which draws inspiration from the human neural system in order to process the information. The goal of MLP is to approximate some mapping functions between input and output vectors [26].

The MLP contains a system of simply interconnected *neurons* which are arranged into at least three layers including an input layer, one or more hidden layers and finally an output layer [27, 33]. The neurons in the input layer do not perform any computation; instead, it serves to pass the input vector to the hidden layers. Each neuron in other layers performs a simple *nonlinear* transformation using an *activation function*, such as rectified linear units (`ReLU`), `tanh` or `sigmoid` function to calculate output of that layer, which enables the MLP to approximate extremely nonlinear functions [26].

The neurons in two consecutive layers are connected by weights θ learned through a training process to approximate the mapping function from input to output vectors. MLP has the learn the mapping function in a supervised manner [26] using a set of training data. Specifically, for a regression problem, the goal of the training process is to approximate the function f such that the derived value of $f(x_i, \theta)$ is close to the actual target value y_i. The difference between the derived and actual target values is considered as an error signal. During training, the error signal is utilised to determine what degree the weights θ in the network should be adjusted in order to reduce the *overall error* of the MLP.

Training a MLP network is normally done using iterative, gradient-based learning [27, 34, 35], e.g. Stochastic Gradient Descent (SGD), Quasi-Newton method, e.g. Limited-memory Broyden–Fletcher–Goldfarb–Shanno (BFGS) so called *L-BFGS* in order to minimise the overall error. Although SGD is easy to implement, optimising/training SGD is difficult with sparse data and low dimensional problems like the HPC. In these cases, L-BFGS is highly competitive or sometimes superior to SGD [34].

It should be noted that training a MLP network has no global convergence guarantee and is sensitive to the initial values of hyperparameters [27], including the activation function, numbers of hidden layers, `hidden_size` - number of neurons in each layer, `solver` - iterative, gradient-based learning, `max_iter` - maximum number of iterations and `alpha` - L2 regularisation parameter which is utilised to prevent overfitting when training the model with the iterative, gradient-based learning [27]. Neural network has been utilised to civil engineering problems since the 1980s [11, 12, 19, 20, 36]. Moreover, many more powerful neural network models have been recently proposed to handle structured data [37, 38]. However, it should be mentioned that this study aims to show the importance of tuning hyperparameters. Classical MLP, therefore, is utilised to make it comparable to previously proposed MLP variants for the HPC problems [12, 19, 20].

2.3. Gradient boosting machines

GBMs or gradient tree boosting originally proposed by Friedman [28] is a boosting machine learning model which utilises a sequences of "weak" or "base" learners with the aim of creating an arbitrarily accurate "strong" learner [28, 29, 39, 40]. A weak learner is defined as one whose performance is at least better than the random guess. In the model, new weak learners are added with the objective of minimising the overall error which also known as the loss of the model.

The GBMs are stagewise additive (ensemble) models in which at a time, a new weak learner is added and trained in order to reduce the overall error of the whole model, and the existing weak learners in the model are not changed [28]. GBMs use regression trees [41] as the weak learners; and iterative, gradient based-learning algorithm, such as SGD, is used to train GBMs in order to minimise the loss when adding weak learners [28]. Specifically, in the first iteration, the algorithm learns the first weak learner, i.e. the first tree, to reduce the overall training error. In the second iteration, the algorithm learns the second tree to reduce the error made by the first tree as demonstrated in Fig. 3. The algorithm repeats this procedure until it builds a decent quality model, such as the loss of the model, i.e. overall error, reaches a desired level. The detailed description of methodology and learning algorithms can be found in the literature, such as, "Greedy function approximation: a gradient boosting machine" [28] and "Gradient boosting machines, a tutorial" [42].

To this end, gradient boosting regressor (GBR) which is GBMs for regression problems, as well as, the Extreme gradient boosting (*XGBoost*) which is a highly scalable extension of GBMs [29] are utilised to handle the presenting HPC regression tasks. It should be noted that XGBoost has been widely recognised and achieved state-of-the-art (SOTA) results in machine learning and data mining challenges [29]. Similar to MLP, the success of gradient based learning in GBMs depends on the initial values of hyperparameters [22] including `n_estimators` - the number of weak learners, i.e. regression trees, `max_deep` - the maximum deep of trees, `loss/objective` - the loss function, and the learning rate. In the next section, the new SOTA performances are illustrated by using GBR and XGBoost with careful hyperparameter initialisation.

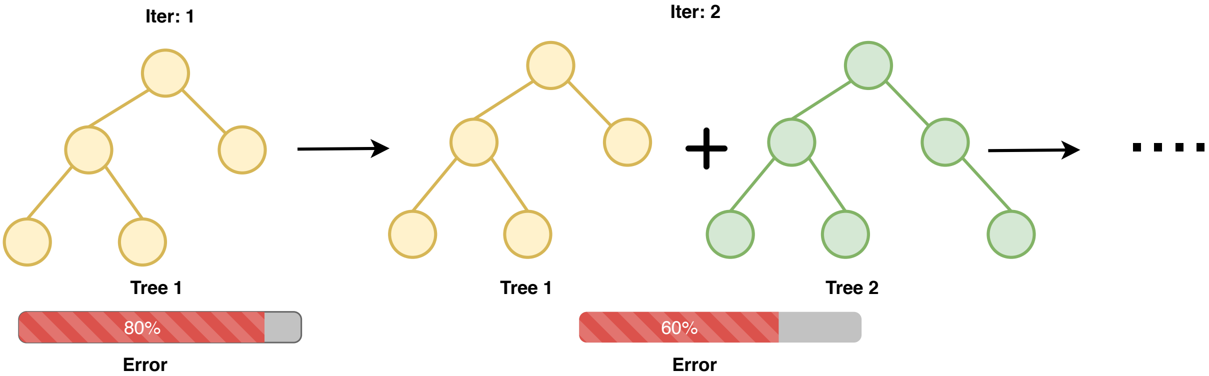


Figure 3: Iteratively learning weak learners (trees) using GBMs in order to reduce the error.

3. High performance concrete data collection and evaluation

3.1. Dataset 1 - Concrete compressive strength

Dataset 1 of concrete compressive strength with a total of 1133 samples is collected at UCI Machine Learning Repository [43, 44]. The statistical details which were extracted and calculated purely from the collected data are presented in Table 1. As can be seen, all the sample data is fulfilled and this is no need to have a procedure to handle missing data.

In order to extract more information regarding the mutual relationship between all input and output features in the dataset, the correlations of features are analysed. This statistical measure is useful as it describes one feature in terms of its association with others. In practice, the observation from this analysis will eventually lead to the choice of the predictive model to be used to maximise the predicting results. Among those available in the literature, Pearson's approach will be used to calculate the correlation coefficient as follows

$$\rho = \frac{\sum \left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sqrt{\sum \left(X_i - \bar{X}\right)^2 \sum \left(Y_i - \bar{Y}\right)^2}} = \frac{E\left[\left(X - \mu_X\right)\left(Y - \mu_Y\right)\right]}{\sigma_X \sigma_Y}, \tag{4}$$

where ρ is the Pearson correlation coefficient. X and Y are two features while overhead bar and subscript i represent the mean value and the i^{th} observation, respectively. Meanwhile, E and σ are the expectation and standard deviation, respectively. The formulation in Eq. (4) ensures the coefficient is bounded by -1 and 1. The value of 0 indicates absolutely no correlation, i.e. no relationship, between a specific pair of features while there will be a perfect positive correlation if the value is 1 or a perfect negative correlation if it is -1. This means that the increase in one quantity leads to the increase (if 1) or decrease (if -1) of the other. If the correlation value goes toward -1 or 1, the association between the features is stronger. An obvious example of a perfect positive correlation is the relationship of a quantity and itself where the correlation coefficient is always 1. On the contrary, the closer the value is to 0, the weaker the correlation gets. It should be noted that, in Pearson

Table 1: Statistics of the datasets

Attribute	Abbreviation	Unit	Minimum	Maximum	Mean	Standard deviation	Missing data
Dataset 1: HPC compressive strength (1133 samples)							
Cement	cmt	kg/m^3	102.00	540.00	276.50	103.47	0%
Blast furnace slag	bfs	kg/m^3	0.00	359.40	74.27	84.25	0%
Fly ash	fash	kg/m^3	0.00	260.00	62.81	71.58	0%
Water	wtr	kg/m^3	121.75	247.00	182.98	21.71	0%
Superplasticizer	sp	kg/m^3	0.00	32.20	6.42	5.80	0%
Coarse aggregate	cagg	kg/m^3	708.00	1145.00	964.83	82.79	0%
Fine aggregate	fagg	kg/m^3	594.00	992.60	770.49	79.37	0%
Age	age	day	1.00	365.00	44.06	60.44	0%
Compressive strength†	fcu	MPa	2.33	82.60	35.84	16.10	0%
Dataset 2: HPC tensile strength (714 samples)							
Cement's compressive strength	fce	MPa	35.50	63.40	50.35	6.80	35%
Cement's tensile strength	fct	MPa	6.90	10.80	8.31	0.66	39%
Curing age	age	day	1.00	388.00	56.73	76.28	0%
Dmax of crushed stone	dmax	mm	12.00	120.00	43.87	26.24	9%
Stone powder content in sand	stnpwd	%	0.00	40.00	10.80	5.56	6%
Fineness modulus of sand	fms	-	2.20	3.55	2.93	0.27	16%
W/B	w/b	-	0.24	1.00	0.45	0.12	1%
Water to cement ratio	w/c	-	0.30	1.43	0.59	0.24	1%
Water	wtr	kg/m^3	70.00	291.00	148.25	33.35	2%
Sand ratio	sndrat	%	24.00	54.00	36.30	6.09	15%
Slump	slm	mm	9.00	260.00	80.27	66.48	11%
Compressive strength	fcu	MPa	4.23	100.50	42.87	22.14	0%
Tensile strength†	fst	MPa	0.35	6.90	3.00	1.36	0%

†Output features. Remainings are input features.

correlation, if two features are independent, the coefficient is close to 0 but not the other way around. This means even though the relationship between quantities is actually strong, their correlation coefficient can still be small.

Fig. 4 presents pair-wise scatter correlation plots and colour map correlation matrix of features of Dataset 1 with correlation coefficient. As can be observed, there is almost no relationship between fine aggregate and fly ash with correlation coefficient of -0.01 while the correlation between water and superplasticizer is fairly strong with the coefficient of -0.59. This is consistent with what has been known in reality.

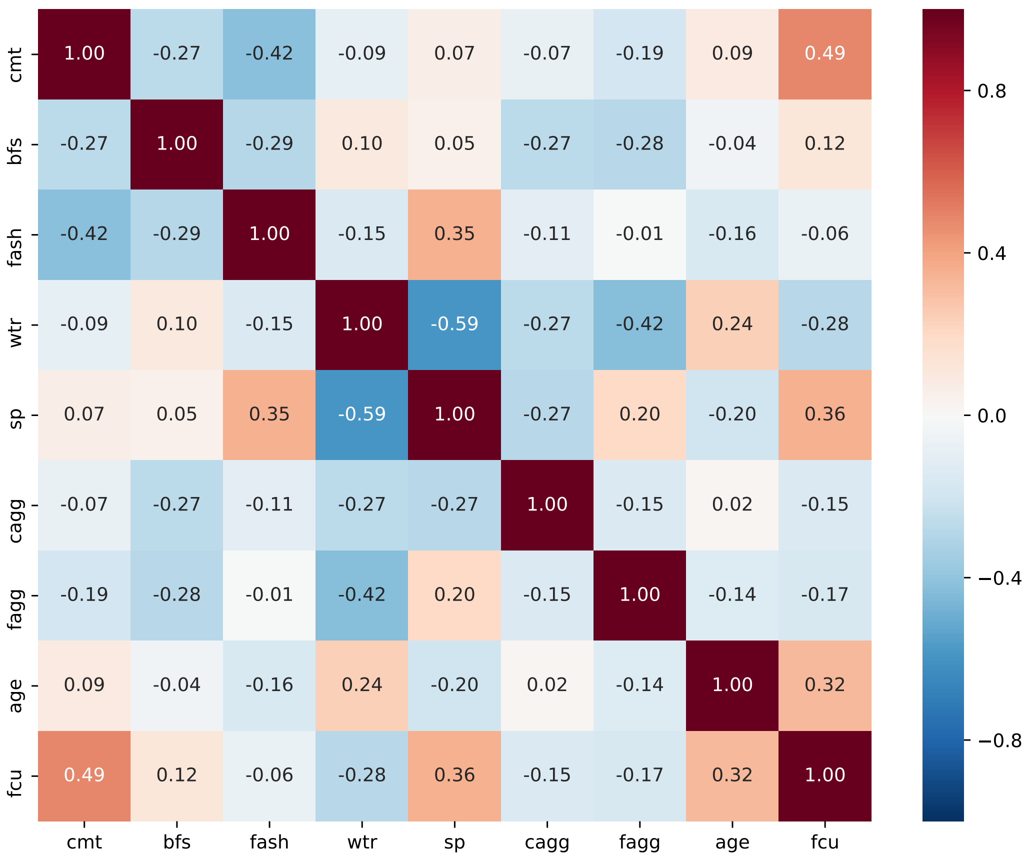


Figure 4: Correlation matrix of features in Dataset 1 with abbreviations of features presented in Table 1.

It is worth to mention that preprocessing data is needed before it is used to train the machine learning models. As the features are not in a uniform scale, they should be normalised to the same range to avoid the training process to be dominated by one or few features with large magnitude. In this study, the process is done by applying the normalisation of features to the range from 0 to 1 before the training This is done, for each of the input features, by dividing each data point by the highest data magnitude in the same feature. Once the models have been trained using normalised data, the predictive results of the output features will be mapped back to its original scale in the test phase.

Another aspect of preprocessing data considered in this study is to generate additional polynomial and interaction features. This is done by considering all or a few polynomial combinations of features in which the maximum degree is predefined. For instance, a pair of features A and B will lead to additional features of $A \times B$, A^2, and B^2 when second order polynomial is defined as the maximum degree. The new feature $A \times B$ is created by the interaction of A and B whereas A^2 and B^2 show no interaction between the two original features. This technique increases the input features by adding new polynomial features which, even though it is not guaranteed, potentially results in better trained models. At the downside, it requires more computational effort and high degrees can cause overfitting. During the training processes in this study, the maximum degree of each original feature will be controlled by the variable `degree` and whether or not only the interaction features, e.g. $A \times B$, are considered depends on the boolean variable `interaction_only`.

3.2. Dataset 2 - Concrete tensile strength

Dataset 2 with 714 samples recording the input and output for tensile strength is shared by Zhao et al. [45]. The statistical details of the dataset are also presented in Table 1. Some of the values in the dataset are given as a range rather than specific values. In those cases, they are replaced by the lower bound before the statistics is carried out. It should be mentioned that, unlike Dataset 1, a considerable amount of missing data can be noticed in Dataset 2. In fact, only two, curing age and compressive strength, out of 12 input features in Dataset 2 are fully collected while the proportions of missing data of the remaining features range from 1% to 39% as shown in Table 1. Previous studies handled this type of issue by removing all the features containing missing data [11, 12, 19]. This practice might not lead to the best performance of the predictive models as only a small portion of data (2 out of 12 input features) was actually used for training the predictive models. Instead, handling the missing data should be performed to make use of all available input features. In this study, the mean imputation which is a common method of imputing missing data is utilised [46, 47]. With this approach, the missing data of a specific feature is replaced by the mean of all available values within that feature. The comparative results which illustrate the advantage of having the missing data filled can be found later in Section 4.2. It is worth mentioning that this study does not aim to compare different methodologies of handling missing data but to shed a light on the need of handling the missing data instead of removing it.

Using the same approach described in the previous section, the correlation matrix for input feature of Dataset 2 of concrete tensile strength with all missing values filled are illustrated in Fig. 5. The plots reveal that, for instance, the mutual relationship between sand ratio and stone powder content in sand is weak while that of tensile strength and compressive strength is quite significant as expected.

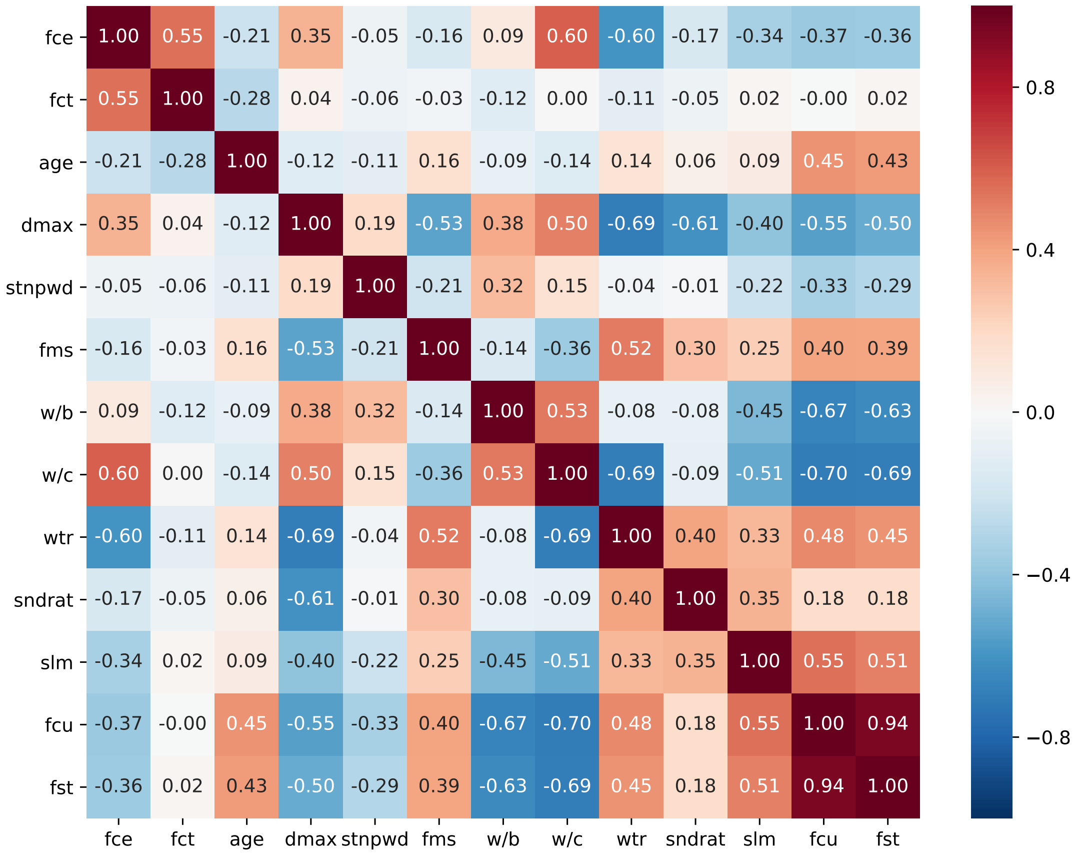


Figure 5: Correlation matrix of features in Dataset 2 with abbreviations of features presented presented in Table 1.

In addition, similar to Dataset 1, the process of data normalisation of all input and output features to the range of 0 to 1 as well as the generation of additional polynomial and interaction features before training will also conducted for Dataset 2.

4. Implementation and results

The training of machine learning models which include SVR, MLP, GBR, and XGBoost is implemented in Python 3.6. The training processes are conducted in the macOS 10.13 platform with the processor of Intel Core i5 CPU 2.9 GHz and memory of 8 GB. The main machine learning and Python libraries used in this work include *scikit-learn* 0.19.1 [30], *XGBoost* 0.80 [29], *NumPy* 1.14.2, *SciPy* 1.0.0 [48], *pandas* 0.23.1 [49]. In order to maintain the randomness in the training processes as well as the reproducibility of the results, the `random_state` parameter is set to 0, where applicable.

For each machine learning model, a random search on hyperparameters is performed to find the best performing model. Note that with the same number of combinations of hyperparameters, random search performs better than grid search and manual search for hyperparameter optimisation [21] as it can be performed in a much larger hyperparameter search space leading to the better performance. In particular, a wide-range preset list of values is defined for each hyperparameter. n combinations of the hyperparameters of each model are then uniformly randomly generated. After that, the model is trained and evaluated with every hyperparameter combination to find the best performing one. In this study, the number of randomly generated combinations n is set to 2000 following the preliminary experiments. The polynomial `degree` $\in \{1, 2, 3, 4\}$. It is observed that `interaction_only`=True for Dataset 1 and `interaction_only`=False for Dataset 2 produced better performances. The choices of this hyperparameter are arbitrarily made during the hyperparameter-tuning process to maintain the balance between the performance of the trained models and the computational costs. The preset list of values of each model is detailed as follows.

For SVR, the hyperparameters are set as follows. `kernel` $\in$ {linear, polynomial, radial basis function (`rbf`), sigmoid}; `C` $\in$ {0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000}; `epsilon`-$\epsilon \in$ {0.01, 0.02, ..., 0.1}; `gamma`-$\gamma \in$ {0.1, 0.2, ..., 1.0}.

For MLP, following [50], the activation function is set to rectified linear units (`ReLU`) as it also produced the best results in the preliminary experiments. Meanwhile, other hyperparameters are set as follows. `number_of_hidden_layers` $\in$ {1, 2}; `hidden_size` $\in$ {100, 200, 300, 400, 500, ..., 1000, 1500, 2000}; `solver` $\in$ {SGD, L-BFGS (`lbfgs`)}; `max_iter` $\in$ {100, 200, 300, 400, 500, ..., 1000}; `alpha` - L2 regularisation parameter $\in$ {0, 0.0001}.

For GBR, `number_of_trees (n_estimators)` $\in$ {100, 200, 500, 1000, 1500, 2000, 2500, 3000, 5000, 10000}; `max_depth` $\in$ {1, 2, ..., 7}; `learning_rate` $\in$ {0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5}; `loss_function (loss)` $\in$ {least squares regression (`ls`), least absolute deviation (`lad`), a combination of the two (`huber`)}.

Similar to GBR, for XGBoost, `number_of_trees (n_estimators)` $\in$ {100, 200, 500, 1000, 1500, 2000, 2500, 3000, 5000, 10000}; `max_depth` $\in$ {1, 2, ..., 7}; `learning_rate` $\in$ {0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5}; `objective_function (objective)` $\in$

{reg:linear, reg:logistic} which are the two objective functions supported by XGBoost for regression problems.

The combination of hyperparameters which produces the best performance on the experimental datasets for each model is reported in Sections 4.1 and 4.2.

With regards to the evaluation of the performance of the machine learning models presented in this study, a set of four indicators are considered including linear correlation coefficient (R) which is related to coefficient of determination (R^2), root mean square error ($RMSE$), mean absolute error (MAE), and mean absolute percentage error ($MAPE$). The calculation of those performance indicators are given as follows

$$R^2 = 1 - \frac{\sum^{n} (y - \hat{y})^2}{\sum^{n} (y - \bar{y})^2}, \tag{5a}$$

$$RMSE = \sqrt{\frac{1}{n} \sum^{n} (y - \hat{y})^2}, \tag{5b}$$

$$MAE = \frac{1}{n} \sum^{n} |y - \hat{y}|, \tag{5c}$$

$$MAPE = \frac{1}{n} \sum^{n} \left| \frac{y - \hat{y}}{y} \right| \times 100, \tag{5d}$$

where y and $\hat{y}$ are actual value and predicted value, respectively, $\bar{y}$ denotes the mean of the actual value and n represents the number of testing data samples. It should be noted that the closer the linear correlation coefficient to 1, the better the prediction. Meanwhile, smaller values of $RMSE$, MAE, and $MAPE$ indicate less error meaning better predictive models are achieved. Apparently, R and $MAPE$ dimensionless while $RMSE$ and MAE have the unit of the output which is `MPa` for both datasets. Among the four performance indicators, $RMSE$ will be mainly used as the representative quantity to discuss the performance of the trained models in this study.

Before the training processes are conducted, the data in each dataset is randomly split into 10 different folds in which the number of samples in each fold is roughly the same. The cross-validation training is then performed by alternately choosing 9 folds to form a training set leaving the remaining fold to be the test set. This process is repeated 10 times until each fold of data is used exactly 9 times for training and 1 time for testing. After training and testing with 10 folds, the means of performance indicators will be calculated to evaluate the trained model. For datasets of small to average size as those used in this study, cross-validation training should be considered to avoid overfitting and to improve the reliability of the training procedures. This also allows effective comparisons being made to existing approaches in the literature [11, 12, 19] where same datasets with similar data settings are used.

4.1. Predictive models for HPC compressive strength

The performance result of the four presenting algorithms (SVR, MLP, GBR, and XGBoost) used to predict the concrete compressive strength is given in Table 2 where the best trained model of each algorithm is highlighted. The outcomes are compared with those reported using the same dataset but different approach and/or input hyperparameters.

As new features for Dataset 1 are added and controlled by `degree` and `interaction_only=True`, the totals of features that are used in the training process are 8, 36, 92, 162 for `degree` = 1, 2, 3, 4, respectively.

As can be observed, with appropriate data preprocessing and hyperparameter tuning, the trained models yield considerably better predictions in terms of both performance and efficiency.

Even though SVR does not yield the best results, it is better than those reported by [11] where SVM algorithm was also involved to build an ensemble model. Particularly, there is a 19% relative improvement in $RMSE$ from 6.17 MPa [11] to 5.00 MPa in this present study with `degree`=1, `kernel`=rbf, `C`=1000, `epsilon`=0.04 and `gamma`=0.5. While R is slightly better, other performance indicators including MAE and $MAPE$ also show an improvement of 11% and 16%, respectively, compared to the referencing work.

Similarly, MLP also gives better prediction than most of the existing results by all performance indicators, except the one reported by [20] where a random train-test data selection is used. Among different neural network architectures tested, the one with 2 hidden layers consisting of 300 and 100 neurons, respectively, yields the best results where additional features have been generated with `degree`=3, `solver`=lbfgs, `max_iter`=1000 and `alpha`=0. Better results can potentially be archived enlarging the network architecture but there will be a computation trade off.

Meanwhile, the performances of GBR and XGBoost witness a significant improvement. In particular, $RMSE$ is reduced approximately by 22% compared to the MFA-ANN [19] where cross validation was performed and by 7% compared to HO-DNN [20]. One can also observe the improvement of the presented models via other performance indicators of R, MAE, and $MAPE$. The GBR performance was achieved with `degree`=1, `n_estimators`=1000, `max_depth`=5, `learning_rate`=0.1 and `loss`=huber. Meanwhile, the performance of XGBoost was achieved with `degree`=1, `n_estimators`=1000, `max_depth`=4, `learning_rate`=0.2 and `objective`=reg:logistic.

Regarding the effectiveness, the current code implementation utilising open-sourced libraries written in Python allows the significant reduction in computational effort compared to the reported results. Indeed, as shown in Table 2, while hundreds of seconds are needed to properly train a model in the works of [12] and [19], implementation of each algorithm in this study only takes a few to tens of seconds to achieve well-trained models with better predictions.

Fig. 6 plots $RMSE$ against different values of `degree` for all four presenting models in comparison with the ensemble approach [11] and ANN [19] in which similar settings of ten-fold cross validation were used. It can be observed that, as `degree` increases, the performance of the trained models may or may not be improved even though in theory the increase would potentially enhance the training process. Despite the results for these particular cases, it is always worth to try different `degree` in the random search for the best performing model.

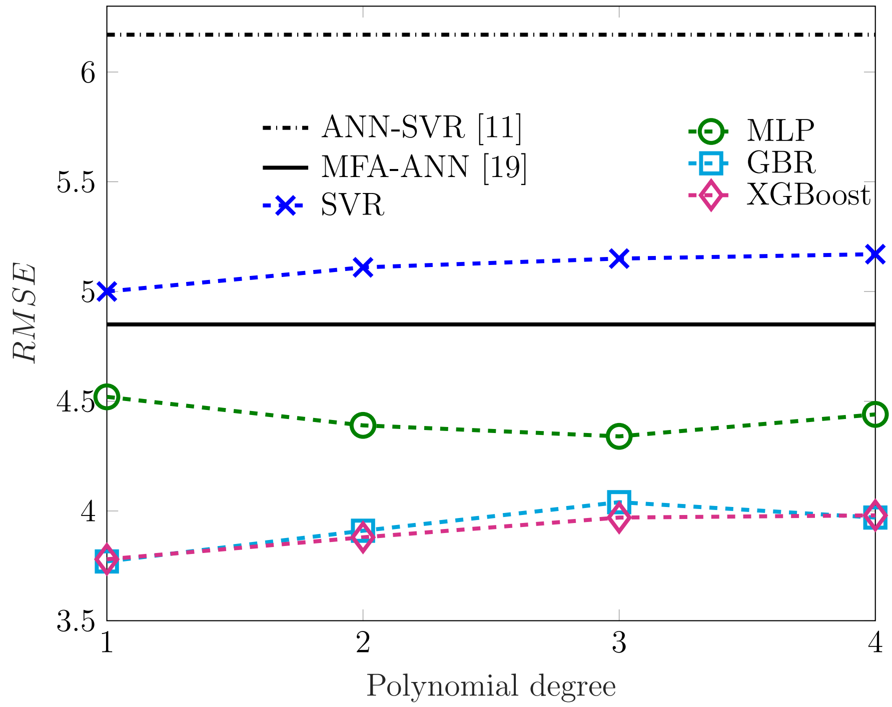


Figure 6: Performance of different methods in prediction of HPC compressive strength with different values of `degree`.

The mean relative feature importance and standard deviation of the inputs of Dataset 1 are given in Fig. 7. This graph made use of the library for XGBoost and it can also be found in the library for GBR. As an interpretation, the higher the value, the more important the feature. This illustration can be used to inform engineers and technicians, the importance of a feature to which they should pay more attention when doing experiments compared to the others. Within limited resources, the information would help to minimise the effects of human errors on the output and eventually the prediction model. Specifically in this case, while the curing age is the feature that has the most significant effect on the outcome of the compressive strength of concrete, the amount of fly ash appears to be the least important input.

The next four figures show the effects of hyperparameters on the performance of the SVR, MLP, GBR, and XGBoost models, respectively. As can be observed from Fig. 8, the $RMSE$ of the prediction by SVR is significantly reduced as a result of the decrease of `epsilon` from 0.25 towards 0, the effect of `C` on the model performance is less pronounced. Besides, the growth of `max_iter` leads to the considerable improvement of the MLP model as plotted in Fig. 9. Meanwhile, Figs. 10 and 11 illustrate the effects of `n_estimators` and `max_depth` on the $RMSE$ predicted by GBR and XGBoost. It is shown that the combination of small values of both hyperparameters may cause large error which becomes minimum when

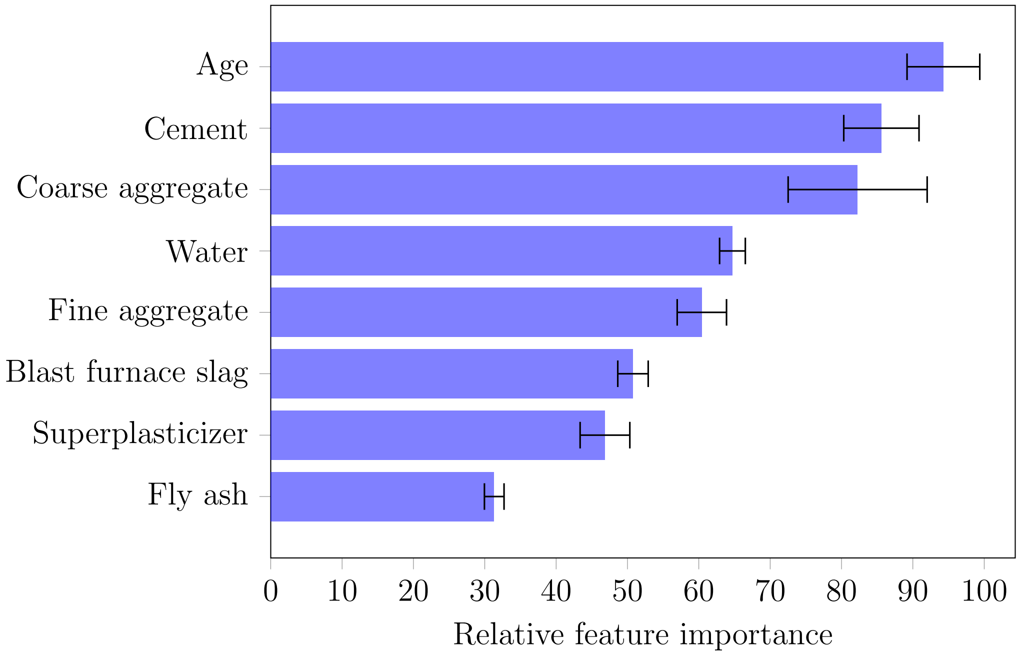


Figure 7: Relative mean and standard deviation of feature importances of data for HPC compressive strength (generated by XGBoost, `degree=1`).

`max_depth` is about 3 or 4. It appears that with `max_depth`≥2, the change of `n_estimators` has less effect on the performance than that of its counterpart. Apparently, the information observed from these figures can be used in the hyperparameter tuning process which is a crucial part of constructing well-performed machine learning models.

Fig. 12 presents the cross validation error ($RMSE$) against `n_estimators` using XGBoost in prediction of compressive strength. For this particular case with the specific setting of other hyperparameters mentioned in the caption of the figure, the increase of `n_estimators` decreases the prediction error which means improvement of the model performance is achieved. At the same time, this figure implies the importance of performing cross validation in order to obtain a reliable predictive model. As can be seen, the prediction error outcome for each of 10 folds can be widely varied with high standard deviation, for instance, from as small as 3.2 MPa to as large as 4.7 MPa for the case of `n_estimators`=1000. Therefore, relying on the result of a single fold may potentially lead to underestimation or overestimation of the prediction error. It should be mentioned that this type of graph is most suitable for the case of `degree=1` which means no additional features are generated apart from the original ones.

4.2. Predictive models for HPC tensile strength

In this part, a similar procedure to those presented in the previous section will be employed to build the prediction models and investigate the effects of hyperparameters on the performances of the predictive models for HPC tensile strength.

The training processes are conducted for two main cases. In the first case, two features

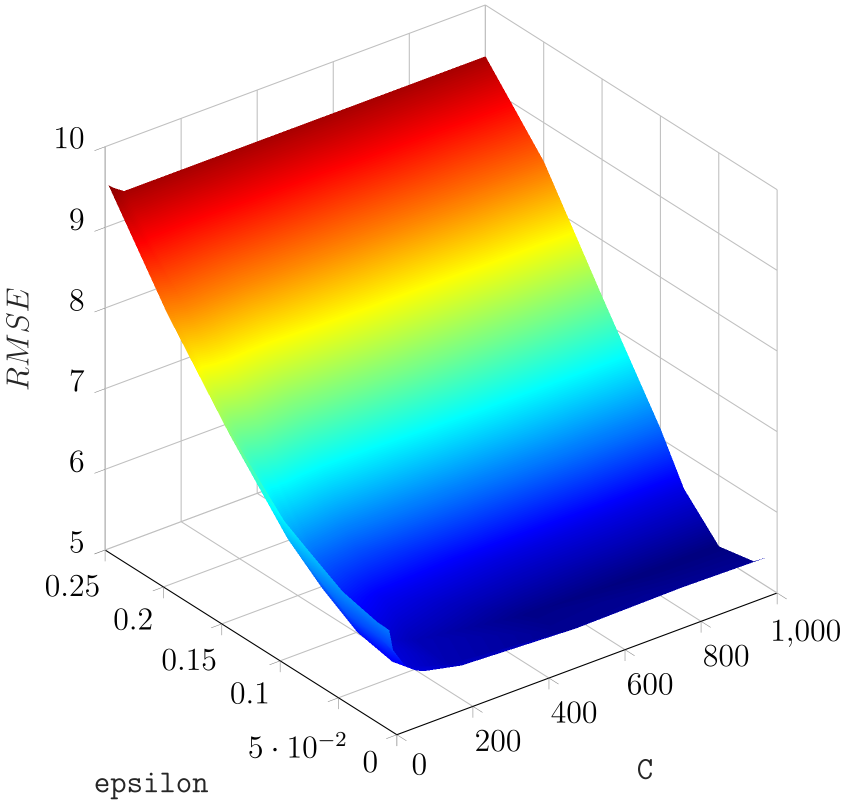


Figure 8: Effects of `C` and `epsilon` on the performance ($RMSE$) of SVR in prediction of compressive strength (`degree`=1, `kernel`='rbf', `gamma`=0.5).

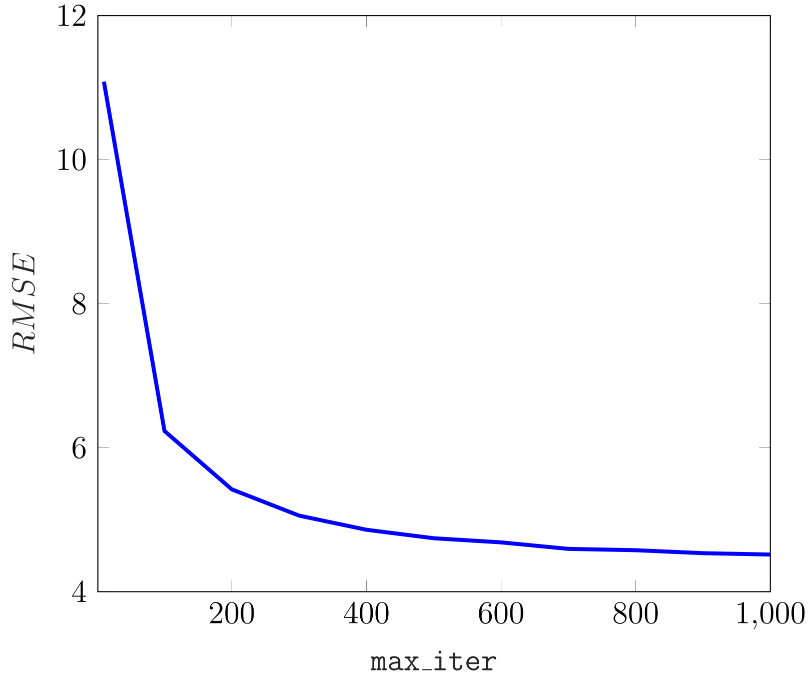


Figure 9: Effects of `max_iter` on the performance ($RMSE$) of MLP in the prediction of compressive strength (`degree`=1, `hidden_layer_sizes`=(300,200), `solver`='lbfgs', `alpha`=0.0001).

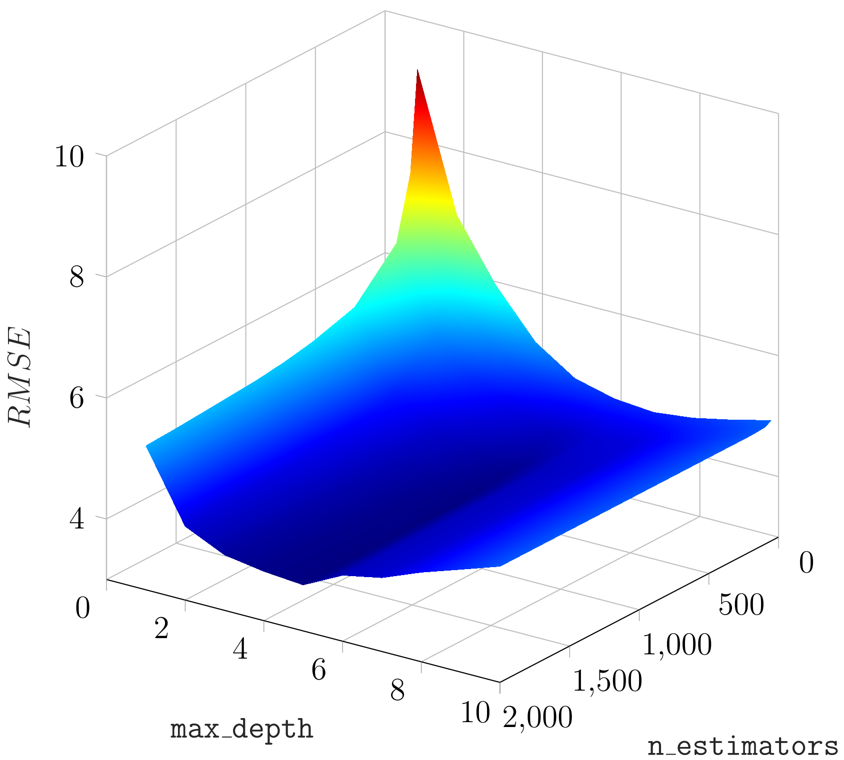


Figure 10: Effects of `n_estimators` and `max_depth` on the performance ($RMSE$) of GBR in the prediction of compressive strength (`degree`=1, `learning_rate`=0.1, `loss`='huber', `min_samples_split`=6).

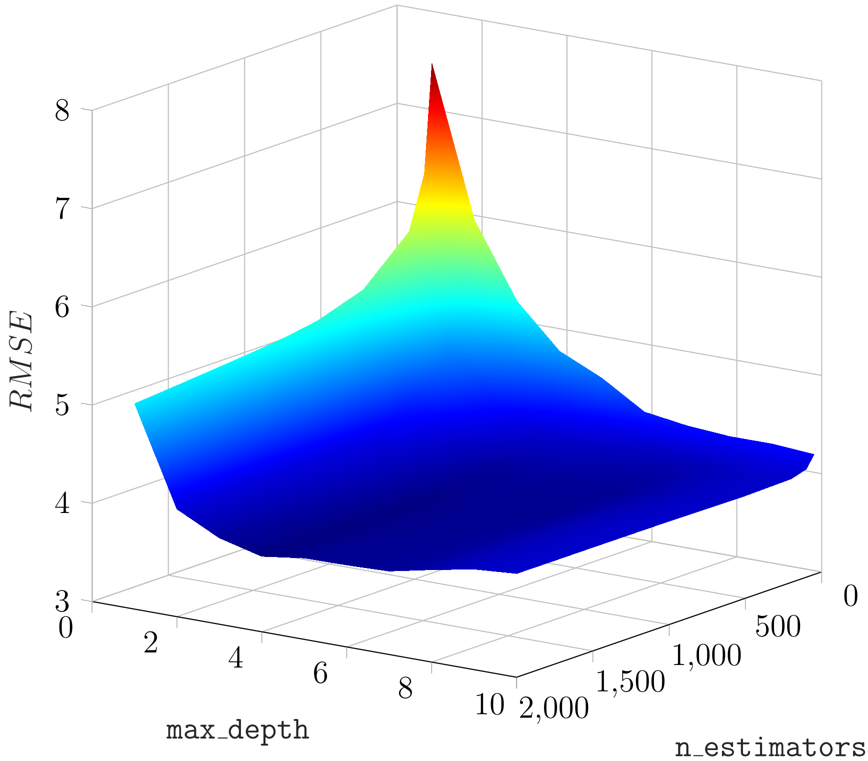


Figure 11: Effects of `n_estimators` and `max_depth` on the performance ($RMSE$) of XGBoost in the prediction of compressive strength (`degree`=1, `learning_rate`=0.2, `objective`='reg:logistic').

Table 2: Comparison of the performance of different methods in prediction of HPC compressive strength

Method	`degree`	Hyperparameter					Performance indicator				Time (s)
							R	$RMSE$	MAE	$MAPE$ (%)	
GEP [51]	1	-	-	-	-	-	0.91	-	5.2	-	-
M-GGP [52]	1	-	-	-	-	-	0.9	7.31	5.48	-	-
ANN-SVR [11]	1	-	-	-	-	-	0.94	6.17	4.24	15.2	-
SFA-LSSVR [12]	1	-	-	-	-	-	0.94	5.62	3.86	12.28	954
MFA-ANN [19]	1	-	-	-	-	-	0.95	4.85	3.41	11.7	276
HO-DNN [20]	1	-	-	-	-	-	0.97	4.05	2.85	-	-
SVR		`kernel`	`C`	`epsilon`	`gamma`	-					
	1	'rbf'	1000	0.04	0.5	-	**0.95**	**5.00**	**3.79**	**12.73**	**28**
	2	'rbf'	100	0.04	0.4	-	0.95	5.11	3.86	12.98	5
	3	'rbf'	100	0.04	0.3	-	0.95	5.15	3.89	13.06	6
	4	'rbf'	100	0.04	0.3	-	0.95	5.17	3.90	13.04	6
MLP		`hidden_layer_sizes`	`solver`	`max_iter`	`alpha`	-					
	1	(300, 200)	'lbfgs'	1000	0.0001	-	0.96	4.52	3.19	10.76	136
	2	(100, 300)	'lbfgs'	1000	0	-	0.96	4.39	2.94	9.7	78
	3	(300, 100)	'lbfgs'	1000	0	-	**0.96**	**4.34**	**2.94**	**9.83**	**89**
	4	(100, 300)	'lbfgs'	1000	0	-	0.96	4.44	3.01	10.1	96
GBR		`n_estimators`	`max_depth`	`learning_rate`	`loss`	`min_samples_split`					
	1	1000	5	0.1	'huber'	6	**0.97**	**3.77**	**2.44**	**8.31**	**29**
	2	1000	4	0.1	'ls'	6	0.97	3.91	2.57	8.86	26
	3	1000	3	0.1	'huber'	2	0.97	4.04	2.66	9.00	60
	4	1000	3	0.1	'huber'	2	0.97	3.97	2.66	8.95	87
XGBoost		`n_estimators`	`max_depth`	`learning_rate`	`objective`	-					
	1	1000	4	0.2	'reg:logistic'	-	**0.97**	**3.78**	**2.47**	**8.64**	**5**
	2	1000	4	0.1	'reg:linear'	-	0.97	3.88	2.57	8.89	19
	3	1000	4	0.1	'reg:logistic'	-	0.97	3.97	2.64	8.95	44
	4	1000	3	0.1	'reg:linear'	-	0.97	3.98	2.62	8.86	53

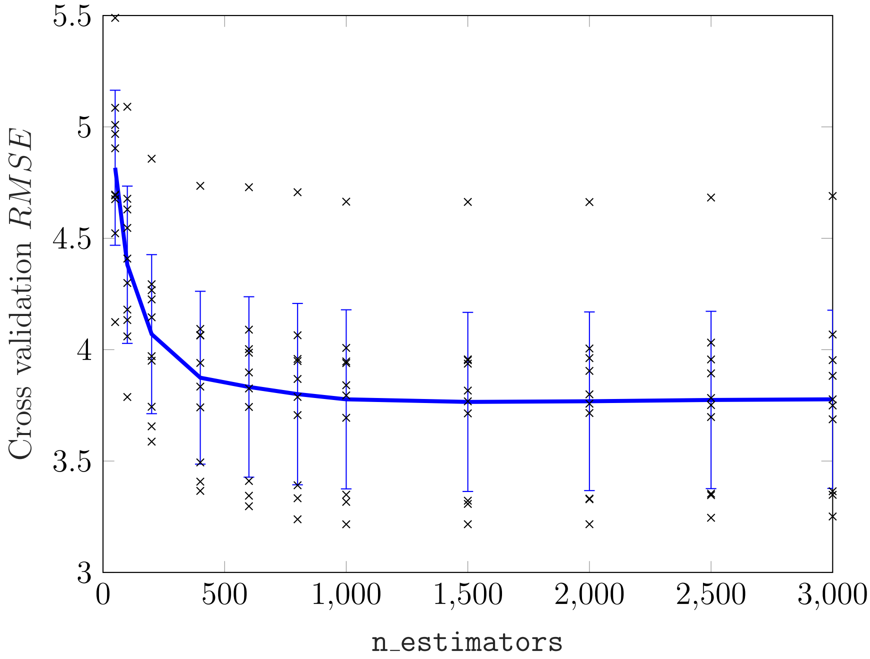


Figure 12: Cross validation error ($RMSE$) on `n_estimators` using XGBoost in prediction of the compressive strength (`degree`=1, `max_depth`=4, `learning_rate`=0.2, `objective`='reg:logistic'). Each black cross indicates a single outcome, the blue line goes through the means, and bars represent standard deviation.

of curing and compressive strength which both contain no missing data are selected as the inputs for training for the prediction of tensile strength. Meanwhile, in the second case, all 12 input features are considered in which those with missing value are filled by the mean of the available data within the feature. It is worth noting that the total number of features actually used in the training process with `interaction_only=False` and `degree` = 1, 2, 3, 4 are 2, 5, 9, 14 for the first case and 12, 90, 454, 1819 for the second case, respectively.

As can be observed from Table 3, the latter significantly reduces $RMSE$ by around 24%-26% when SVR and MLP are employed. Meanwhile, GBR and XGBoost enable even higher improvement of around 30% in $RMSE$ compared to the former case and the recent work of [19] in which the same two features were also used. This remarkable improvement is illustrated in Fig. 13 where $RMSE$ is plotted with respect to polynomial degree, i.e. `degree` parameter. This graph also reveals the benefit of using higher order polynomials in Dataset 2 to generate additional input features and ultimately obtain better prediction models.

The feature importance printed in Fig. 14 indicates that the input feature of concrete compressive strength (fcu) has the highest influence in the prediction of the output of concrete tensile strength. On the contrary, the tensile strength of cement (fct) remains the least important input feature.

Fig. 15 illustrates the effect of `epsilon` and `C` hyperparameters on the SVR model performance indicator of $RMSE$ while the relation of `max_iter` and $RMSE$ of MLP model is shown in Fig. 16. Additionally, Figs. 17 and 18 describe the effect of `n_estimators`

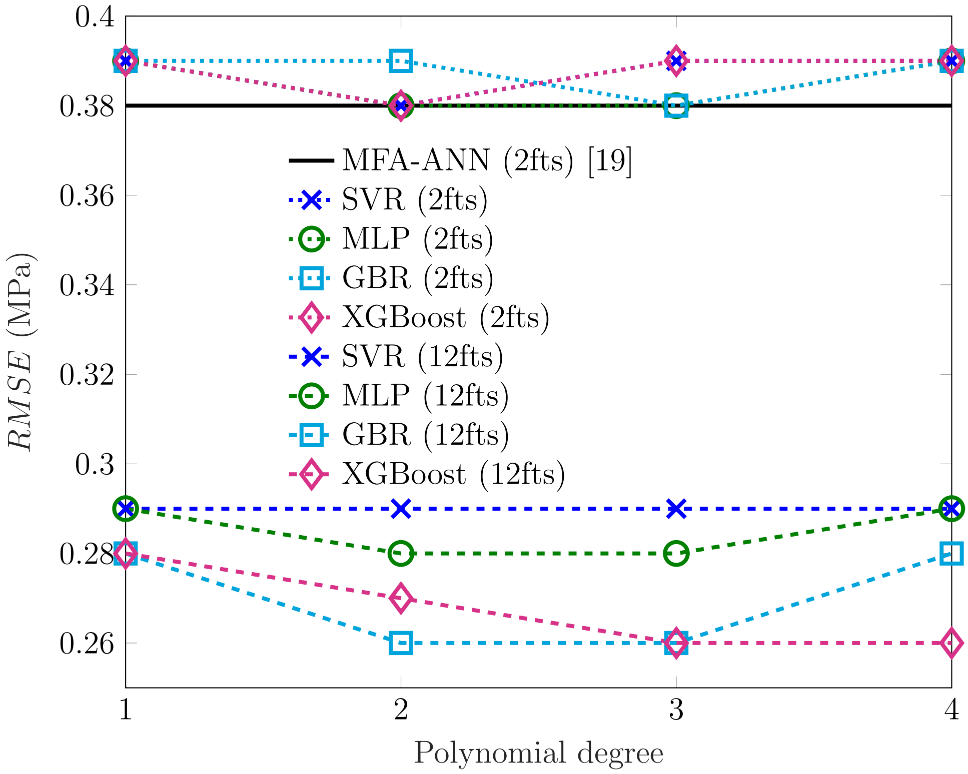


Figure 13: Performance of different methods in prediction of HPC tensile strength with different values of `degree`.

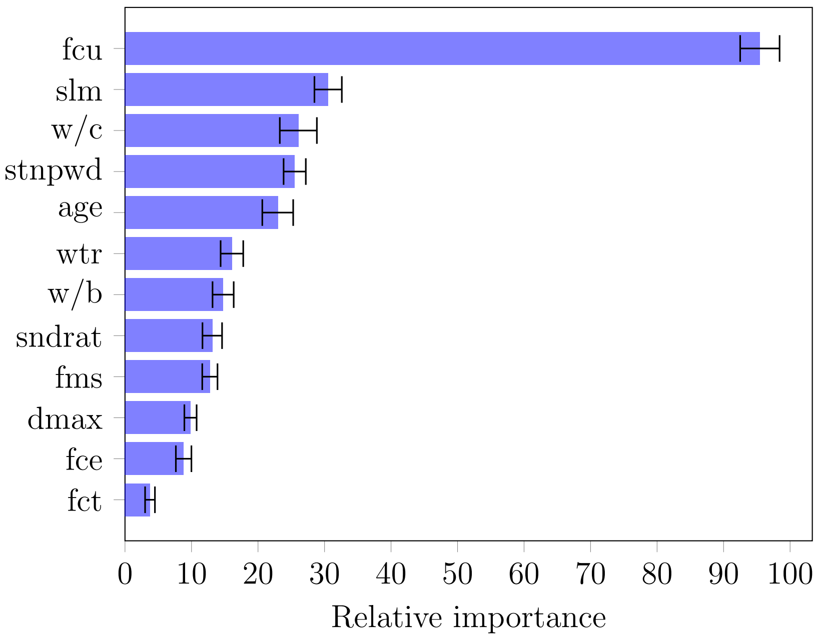


Figure 14: Relative mean and standard deviation of feature importances of data for HPC tensile strength (generated by XGBoost, `degree`=1).

and `max_depth` on the performance of the GBR and XGBoost, respectively. Similar to the previous section using Dataset 1, the variations of the model performance $RMSE$ with respect to the changes of different hyperparameters in this case of Dataset 2 are not much different across all four models used in this study. Meanwhile, the plot of `n_estimators` - RMSE relation for XGBoost model with `degree=1` in Fig. 19 indicates that the increase of this hyperparameter does not guarantee better performance even though it always leads to higher computational effort. In this particular setting of the problem, `n_estimators`=400 yields the lowest $RMSE$ meaning the best prediction. This is consistent with those are shown in Table 3.

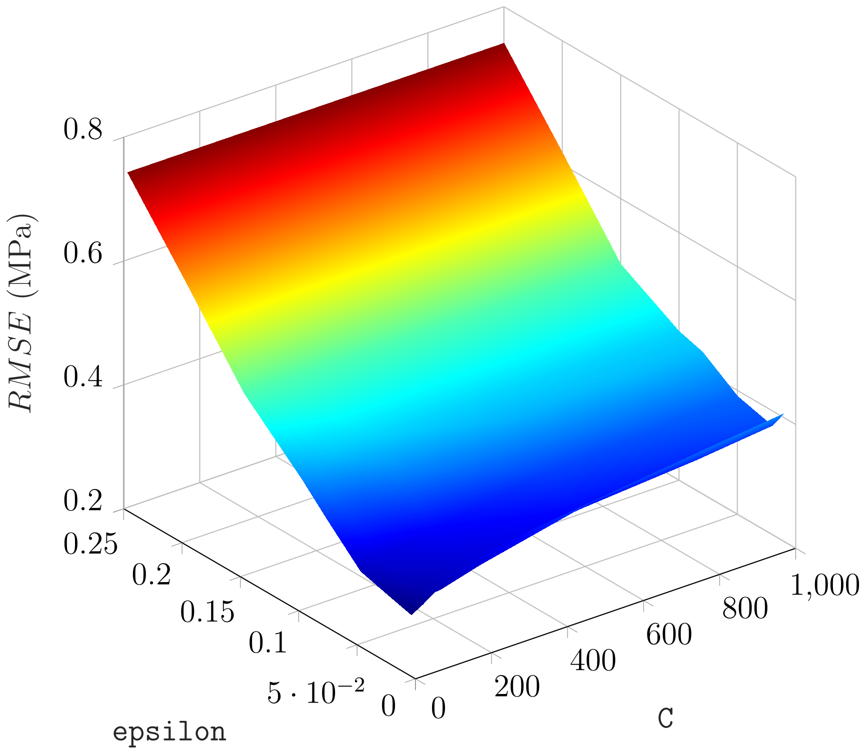


Figure 15: Effects of `C` and `epsilon` on the performance ($RMSE$) of SVR in the prediction of tensile strength (`degree`=1, `kernel`='rbf', `gamma`=0.9).

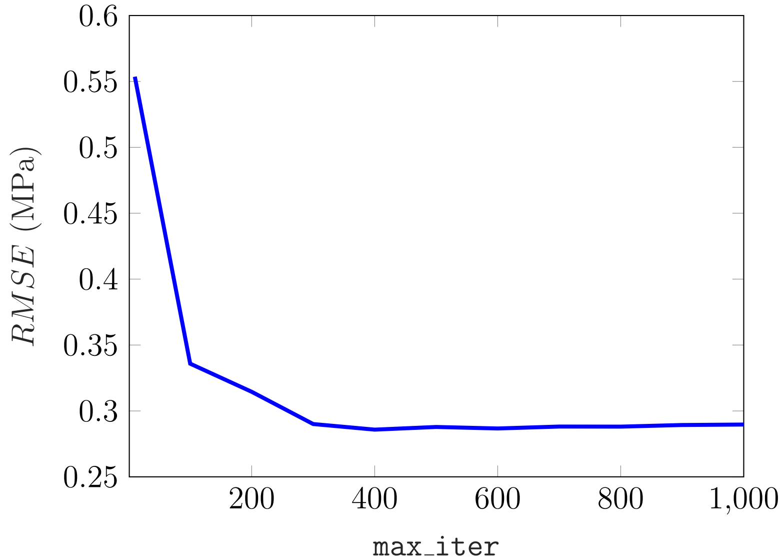


Figure 16: Effects of `max_iter` on the performance ($RMSE$) of MLP in prediction of tensile strength (`degree`=1, `hidden_layer_sizes`=(100,100), `solver`='lbfgs', `alpha`=0).

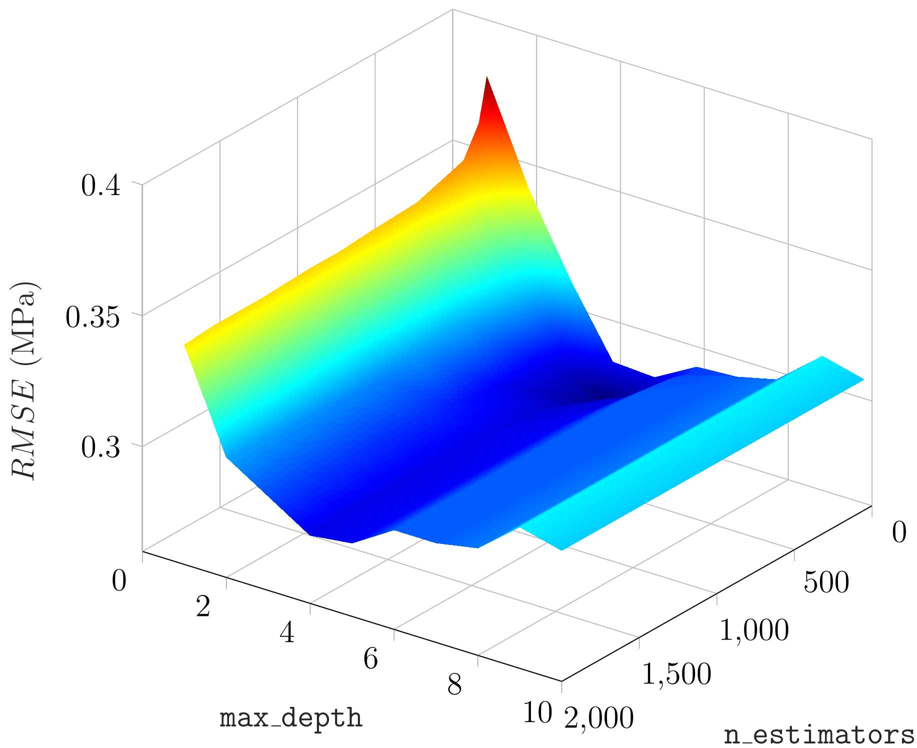


Figure 17: Effects of `n_estimators` and `max_depth` on the performance ($RMSE$) of GBR in the prediction of tensile strength (`degree`=1, `learning_rate`=0.02, `loss`='huber', `min_samples_split`=3).

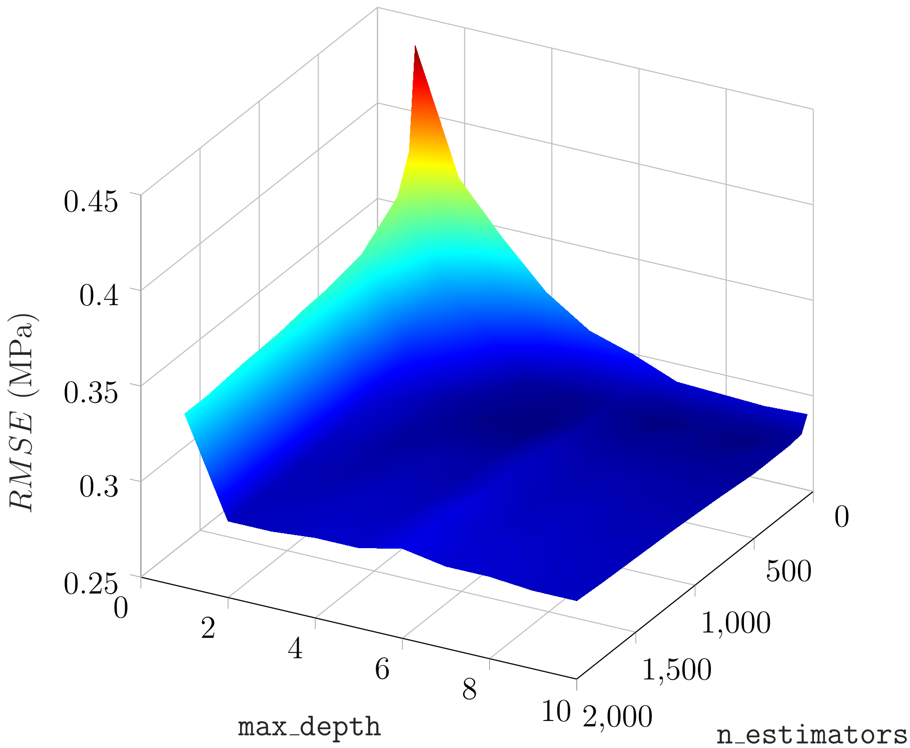


Figure 18: Effects of `n_estimators` and `max_depth` on the performance ($RMSE$) of XGBoost in prediction of tensile strength (`degree`=1, `learning_rate`=0.01, `objective`='reg:logistic').

Table 3: Comparison of the performance of different methods in prediction of HPC tensile strength

Method	# features	`degree`	Hyperparameter					Performance indicator				Time (s)
								R	$RMSE$	MAE	$MAPE$ (%)	
Fitting curve [53]	2	1	-	-	-	-	-	0.93	0.45	0.35	15.99	-
MFA-ANN [19]	2	1	-	-	-	-	-	0.96	0.38	0.28	10.59	276
SVR			`kernel`	`C`	`epsilon`	`gamma`	-					
	2	1	'rbf'	5000	0.03	0.9	-	0.96	0.39	0.27	10.81	9
		2	'rbf'	2000	0.03	0.9	-	**0.96**	**0.38**	**0.27**	**10.64**	**6**
		3	'rbf'	5000	0.03	0.3	-	0.96	0.39	0.27	10.69	7
		4	'rbf'	2000	0.02	0.2	-	0.96	0.39	0.27	10.53	3
	12	1	'rbf'	20	0.01	0.9	-	**0.98**	**0.29**	**0.20**	**7.90**	**2**
		2	'rbf'	10	0.01	0.4	-	0.98	0.29	0.20	7.96	2
		3	'rbf'	10	0.02	0.2	-	0.98	0.29	0.20	8.54	3
		4	'rbf'	10	0.02	0.1	-	0.98	0.29	0.21	8.67	8
MLP			`hidden_layer_sizes`	`solver`	`max_iter`	`alpha`	-					
	2	1	(300, 300)	'lbfgs'	1000	0.0001	-	0.96	0.39	0.28	10.52	86
		2	(200, 100)	'lbfgs'	400	0.0001	-	0.96	0.38	0.27	10.32	14
		3	(100, 200)	'lbfgs'	1000	0.0001	-	**0.96**	**0.38**	**0.27**	**10.15**	**27**
		4	(200, 100)	'lbfgs'	400	0.0001	-	0.96	0.39	0.27	10.37	9
	12	1	(100, 100)	'lbfgs'	1000	0	-	0.98	0.29	0.20	8.00	26
		2	(300, 300)	'lbfgs'	200	0.0001	-	**0.98**	**0.28**	**0.19**	**8.06**	**35**
		3	(100, 300)	'lbfgs'	300	0.0001	-	0.98	0.28	0.20	8.01	29
		4	(300, 100)	'lbfgs'	100	0.0001	-	0.98	0.29	0.21	8.60	72
GBR			`n_estimators`	`max_depth`	`learning_rate`	`loss`	`min_samples_split`					
	2	1	200	3	0.02	'huber'	3	0.96	0.39	0.27	10.68	3
		2	100	3	0.05	'huber'	5	0.96	0.39	0.27	10.58	2
		3	100	3	0.05	'huber'	5	**0.96**	**0.38**	**0.27**	**10.45**	**2**
		4	500	2	0.02	'huber'	3	0.96	0.39	0.27	10.51	5
	12	1	100	4	0.2	'huber'	6	0.98	0.28	0.19	7.06	2
		2	500	3	0.1	'huber'	4	0.98	0.26	0.18	6.89	17
		3	1000	2	0.1	'huber'	6	**0.98**	**0.26**	**0.18**	**6.80**	**94**
		4	500	2	0.1	'huber'	5	0.98	0.28	0.18	7.23	229
XGBoost			`n_estimators`	`max_depth`	`learning_rate`	`objective`	-					
	2	1	500	4	0.01	'reg:logistic'	-	0.96	0.39	0.28	11.08	1
		2	100	2	0.2	'reg:logistic'	-	**0.96**	**0.38**	**0.28**	**10.67**	**1**
		3	500	3	0.02	'reg:logistic'	-	0.96	0.39	0.28	10.63	2
		4	200	4	0.05	'reg:logistic'	-	0.96	0.39	0.28	10.55	1
	12	1	400	5	0.1	'reg:logistic'	-	0.98	0.28	0.18	7.02	3
		2	1000	4	0.1	'reg:logistic'	-	**0.98**	**0.27**	**0.17**	**6.59**	**22**
		3	1000	2	0.1	'reg:linear'	-	0.98	0.27	0.18	6.97	66
		4	1000	4	0.05	'reg:linear'	-	0.98	0.27	0.18	6.83	544

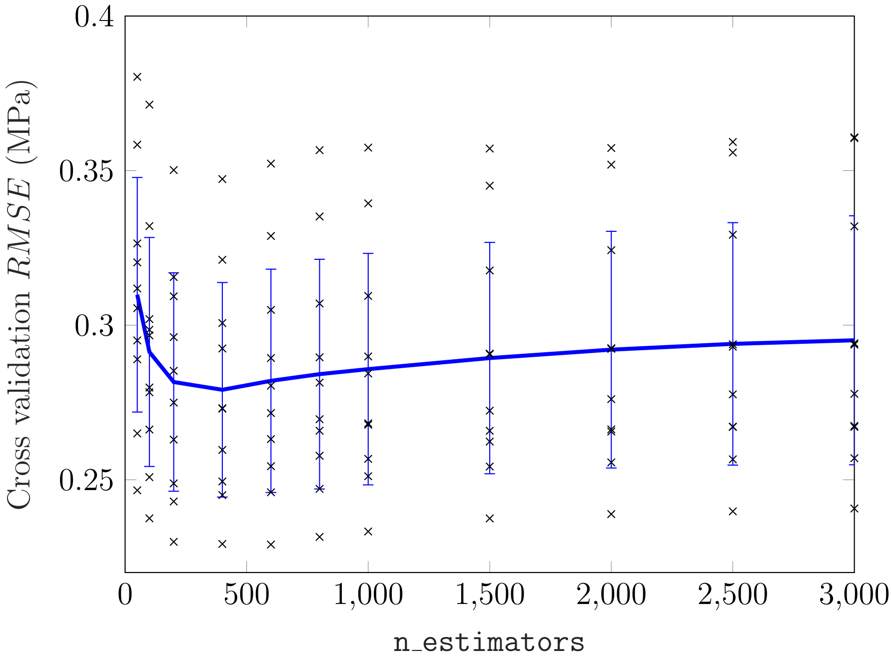


Figure 19: Cross validation error ($RMSE$) on `n_estimators` using XGBoost in prediction of tensile strength (`degree`=1, `max_depth`=5, `learning_rate`=0.1, `objective`='reg:logistic'). Each black cross indicates single outcome, blue line goes through the means, and bars represent standard deviation.

5. Concluding remarks

Four machine learning algorithms including SVR, MLP, GBR, and XGBoost are employed to predict the compressive and tensile strengths of HPC in this study. Open-sourced machine learning libraries are involved in the implementation which enhances the model performance and significantly speeds up the running process. This allows the random search to be conducted in the hyperparameter tuning process in which a much larger search space is considered with the same computational effort. The comparative studies reveal the effects of some hyperparameters on the performance of each model. It is shown that GBR and XGBoost yield better prediction results with significantly less computational effort compared to that of SVR and MLP. Also, by using the single mean imputation method, the handling of missing data in the dataset of concrete tensile strength enables the use of all 12 input features which gives considerably better prediction results compared to the case where two fully collected features are employed. The drawbacks of the current approach include the time-consuming process of parameter tuning and the reliance of the quality of the datasets. The former can be mitigated by using a optimisation algorithm, e.g. Genetic Algorithm, to automatise the tuning process in which the variables are the hyperparameters and the objective function is minimisation of the prediction errors. Meanwhile, the quality of the datasets can be controlled by careful processes of experiment design, test, and measurement. In general, the approach presented in this study can be applied to other engineering datasets where input and output features are clearly defined. In addition, the speed of the training process presented in this study can be improved by using a fully scalable implementation that can be run in parallel processors. With an aim to assist interested readers to get familiar with the implementation of machine learning models and reproduce the results presented in this study, the developed codes are made open-sourced at https://github.com/hoangnguyence/hpconcrete.

References

References

[1] A. Neville, P.-C. Aïtcin, High performance concrete—An overview, Materials and Structures 31 (2) (1998) 111–117.

[2] C. K. Y. Leung, Concrete as a Building Material, in: Encyclopedia of Materials: Science and Technology, Elsevier, 2001, pp. 1471–1479.

[3] H. Adeli, Four Decades of Computing in Civil Engineering, in: CIGOS 2019, Innovation for Sustainable Infrastructure, Lecture Notes in Civil Engineering, Springer, 2019, pp. 3–11.

[4] T. N. Nguyen, S. Lee, H. Nguyen-Xuan, J. Lee, A novel analysis-prediction approach for geometrically nonlinear problems using group method of data handling, Computer Methods in Applied Mechanics and Engineering 354 (2019) 506–526. `doi:10.1016/j.cma.2019.05.052`.

[5] S. Lee, J. Ha, M. Zokhirova, H. Moon, J. Lee, Background Information of Deep Learning for Structural Engineering, Archives of Computational Methods in Engineering 25 (1) (2018) 121–129. `doi:10.1007/s11831-017-9237-0`.

[6] H.-G. Ni, J.-Z. Wang, Prediction of compressive strength of concrete by neural networks, Cement and Concrete Research 30 (8) (2000) 1245–1250.

[7] M. H. Rafiei, W. H. Khushefati, R. Demirboga, H. Adeli, Novel Approach for Concrete Mixture Design Using Neural Dynamics Model and Virtual Lab Concept, Materials Journal 114 (1) (2017) 117–127.

[8] M. H. Rafiei, W. H. Khushefati, R. Demirboga, H. Adeli, Supervised Deep Restricted Boltzmann Machine for Estimation of Concrete, Materials Journal 114 (2) (2017) 237–244.
[9] I.-C. Yeh, L.-C. Lien, Knowledge discovery of concrete material using Genetic Operation Trees, Expert Systems with Applications 36 (3, Part 2) (2009) 5807–5812.
[10] Chou Jui-Sheng, Chiu Chien-Kuo, Farfoura Mahmoud, Al-Taharwa Ismail, Optimizing the Prediction Accuracy of Concrete Compressive Strength Based on a Comparison of Data-Mining Techniques, Journal of Computing in Civil Engineering 25 (3) (2011) 242–253.
[11] J.-S. Chou, A.-D. Pham, Enhanced artificial intelligence for ensemble approach to predicting high performance concrete compressive strength, Construction and Building Materials 49 (2013) 554–563.
[12] J.-S. Chou, W. K. Chong, D.-K. Bui, Nature-Inspired Metaheuristic Regression System: Programming and Implementation for Civil Engineering Applications, Journal of Computing in Civil Engineering 30 (5) (2016) 04016007.
[13] T. Le-Duc, Q.-H. Nguyen, H. Nguyen-Xuan, Balancing composite motion optimization, Information Sciences 520 (2020) 250–270. `doi:10.1016/j.ins.2020.02.013`.
[14] M. Engen, M. A. N. Hendriks, J. Kohler, J. A. Overli, E. Aldstedt, E. Mortsell, O. Sæter, R. Vigre, Predictive strength of ready-mixed concrete: Exemplified using data from the Norwegian market, Structural Concrete 19 (3) (2018) 806–819.
[15] H. I. Erdal, O. Karakurt, E. Namli, High performance concrete compressive strength forecasting using ensemble models based on discrete wavelet transform, Engineering Applications of Artificial Intelligence 26 (4) (2013) 1246–1254.
[16] S. Czarnecki, L. Sadowski, J. Hola, Artificial neural networks for non-destructive identification of the interlayer bonding between repair overlay and concrete substrate, Advances in Engineering Software 141 (2020) 102769. `doi:10.1016/j.advengsoft.2020.102769`.
[17] A. Dey, G. Miyani, A. Sil, Application of artificial neural network (ANN) for estimating reliable service life of reinforced concrete (RC) structure bookkeeping factors responsible for deterioration mechanism, Soft Computing 24 (3) (2020) 2109–2123. `doi:10.1007/s00500-019-04042-y`.
[18] A. Falih, A. Z. M. Shammari, Hybrid constrained permutation algorithm and genetic algorithm for process planning problem, Journal of Intelligent Manufacturing 31 (5) (2020) 1079–1099. `doi:10.1007/s10845-019-01496-7`.
[19] D.-K. Bui, T. Nguyen, J.-S. Chou, H. Nguyen-Xuan, T. D. Ngo, A modified firefly algorithm-artificial neural network expert system for predicting compressive and tensile strength of high-performance concrete, Construction and Building Materials 180 (2018) 320–333.
[20] T. Nguyen, A. Kashani, T. Ngo, S. Bordas, Deep neural network with high-order neuron for the prediction of foamed concrete strength, Computer-Aided Civil and Infrastructure Engineering 34 (4) (2018) 316–332.
[21] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research 13 (2012) 281–305.
[22] P. Probst, A.-L. Boulesteix, B. Bischl, Tunability: Importance of hyperparameters of machine learning algorithms., Journal of Machine Learning Research 20 (53) (2019) 1–32.
[23] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, Berlin, Heidelberg, 1995.
[24] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[25] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and Computing 14 (3) (2004) 199–222.
[26] M. Gardner, S. Dorling, Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences, Atmospheric Environment 32 (14-15) (1998) 2627–2636.
[27] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, 2016.
[28] J. H. Friedman, Greedy function approximation: a gradient boosting machine, Annals of statistics (2001) 1189–1232.
[29] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, 2016, pp. 785–794.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[31] V. Vapnik, S. E. Golowich, A. Smola, Support vector method for function approximation, regression estimation and signal processing, in: Proceedings of the 9th International Conference on Neural Information Processing Systems, NIPS'96, MIT Press, 1996, pp. 281–287.

[32] J.-S. Chou, A.-D. Pham, Smart artificial firefly colony algorithm-based support vector regression for enhanced forecasting in civil engineering, Comp.-Aided Civil and Infrastruct. Engineering 30 (9) (2015) 715–732.

[33] N. Siddique, H. Adeli, Computational intelligence: synergies of fuzzy logic, neural networks and evolutionary computing, John Wiley & Sons, 2013.

[34] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, A. Y. Ng, On optimization methods for deep learning, in: Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11, 2011, pp. 265–272.

[35] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning Internal Representations by Error Propagation, MIT Press, 1986, p. 318–362.

[36] H. Adeli, C. Yeh, Perceptron learning in engineering design, Computer-Aided Civil and Infrastructure Engineering 4 (4) (1989) 247–256.

[37] M. Ahmadlou, H. Adeli, Enhanced probabilistic neural network with local decision circles: A robust classifier, Integrated Computer-Aided Engineering 17 (3) (2010) 197–210.

[38] M. H. Rafiei, H. Adeli, A new neural dynamic classification algorithm, IEEE transactions on neural networks and learning systems 28 (12) (2017) 3074–3083.

[39] R. E. Schapire, A brief introduction to boosting, in: Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'99, Morgan Kaufmann Publishers Inc., 1999, pp. 1401–1406.

[40] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, 3rd Edition, Morgan Kaufmann Publishers Inc., 2011.

[41] L. Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone, Classification and Regression Trees, Wadsworth and Brooks, 1984.

[42] A. Natekin, A. Knoll, Gradient boosting machines, a tutorial, Frontiers in neurorobotics 7 (2013) 21.

[43] I.-C. Yeh, UCI Machine Learning Repository: Concrete Compressive Strength Data Set (1998). URL `https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength`

[44] I.-C. Yeh, UCI Machine Learning Repository: Concrete Slump Test Data Set (2008). URL `https://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test`

[45] S. Zhao, F. Hu, X. Ding, M. Zhao, C. Li, S. Pei, Dataset of tensile strength development of concrete with manufactured sand, Data in Brief 11 (2017) 469–472.

[46] R. L. Brown, Efficacy of the indirect approach for estimating structural equation models with missing data: A comparison of five methods, Structural Equation Modeling: A Multidisciplinary Journal 1 (4) (1994) 287–316.

[47] H. Kang, The prevention and handling of the missing data, Korean journal of anesthesiology 64 (5) (2013) 402.

[48] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python (2001–). URL `http://www.scipy.org/`

[49] W. McKinney, Data structures for statistical computing in python, in: Proceedings of the 9th Python in Science Conference, ACM, 2010, pp. 51–56.

[50] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th international conference on machine learning, 2010, pp. 807–814.

[51] S. M. Mousavi, P. Aminian, A. H. Gandomi, A. H. Alavi, H. Bolandi, A new predictive model for compressive strength of HPC using gene expression programming, Advances in Engineering Software 45 (1) (2012) 105–114.

[52] A. H. Gandomi, A. H. Alavi, D. M. Shadmehri, M. G. Sahab, An empirical model for shear capacity

of RC deep beams using genetic-simulated annealing, Archives of Civil and Mechanical Engineering 13 (3) (2013) 354–369.

[53] S. Zhao, X. Ding, M. Zhao, C. Li, S. Pei, Experimental study on tensile strength development of concrete with manufactured sand, Construction and Building Materials 138 (2017) 247–253.