applied sciences

MDPI

Article

A Machine Learning Approach to Prediction of the Compressive Strength of Segregated Lightweight Aggregate Concretes Using Ultrasonic Pulse Velocity

Violeta Migallón [1,*,†], Héctor Penadés [1,†], José Penadés [1,†] and Antonio José Tenza-Abril [2,†]

[1] Department of Computer Science and Artificial Intelligence, University of Alicante, 03071 Alicante, Spain
[2] Department of Civil Engineering, University of Alicante, 03071 Alicante, Spain
* Correspondence: violeta@ua.es
† These authors contributed equally to this work.

Abstract: Lightweight aggregate concrete (LWAC) is an increasingly important material for modern construction. However, although it has several advantages compared with conventional concrete, it is susceptible to segregation due to the low density of the incorporated aggregate. The phenomenon of segregation can adversely affect the mechanical properties of LWAC, reducing its compressive strength and its durability. In this work, several machine learning techniques are used to study the influence of the segregation of LWAC on its compressive strength, including the K-nearest neighbours (KNN) algorithm, regression tree-based algorithms such as random forest (RF) and gradient boosting regressors (GBRs), artificial neural networks (ANNs) and support vector regression (SVR). In addition, a weighted average ensemble (WAE) method is proposed that combines RF, SVR and extreme GBR (or XGBoost). A dataset that was recently used for predicting the compressive strength of LWAC is employed in this experimental study. Two different types of lightweight aggregate (LWA), including expanded clay as a coarse aggregate and natural fine limestone aggregate, were mixed to produce LWAC. To quantify the segregation in LWAC, the ultrasonic pulse velocity method was adopted. Numerical experiments were carried out to analyse the behaviour of the obtained models, and a performance improvement was shown compared with the machine learning models reported in previous works. The best performance was obtained with GBR, XGBoost and the proposed weighted ensemble method. In addition, a good choice of weights in the WAE method allowed our approach to outperform all of the other models.

Keywords: concrete; lightweight aggregate; prediction; compressive strength; machine learning; average weighted ensemble

Citation: Migallón, V.; Penadés, H.; Penadés, J.; Tenza-Abril, A.J. A Machine Learning Approach to Prediction of the Compressive Strength of Segregated Lightweight Aggregate Concretes Using Ultrasonic Pulse Velocity. *Appl. Sci.* **2023**, *13*, 1953. https://doi.org/10.3390/app13031953

Academic Editors: Maria Sozanska, Mariusz Jaśniok and Zbigniew Perkowski

Received: 11 January 2023
Revised: 26 January 2023
Accepted: 30 January 2023
Published: 2 February 2023

1. Introduction

One of the most commonly used building materials in civil and structural engineering is concrete. However, the use of conventional concrete as a basic building material is facing growing challenges. The expansion of the construction industry has created significant demand for lightweight, high-strength structures for high-rise buildings, long-span bridges, and offshore platforms. Lightweight aggregate concrete (LWAC) contains light aggregates (LWAs), such as expanded clay, shale, or slate [1], and is an important and widely used material in advanced architecture, with a low density as the most obvious characteristic that distinguishes it from conventional concrete. LWAC stands out as an extremely versatile building material due to the wide range of potentially available LWAs. Over the last two decades, there has been growing interest in reducing the density of concrete while maintaining as much strength as possible [2]. LWAC also has certain advantages compared with conventional concrete, such as low thermal conductivity, better fire resistance, and insulation against heat and sound. Compared with conventional concrete, LWAC can effectively reduce dead load (thus reducing the costs due to the quantity of structural steel

needed and the foundation size [3]) and can improve the seismic capacity of buildings [4]. Improving the performance of concrete is crucial in order to enable sustainable development. The rheological performance of fresh concrete is characterised by diversity and complexity, and fresh concrete must have sufficient workability to ensure the quality of concrete constructions [5]. In addition, good compaction is critical to ensure that the material fills all the voids in the formworks. When fresh LWAC is placed in the desired area, it must be compacted to remove entrapped air and voids, and to consolidate it. However, over-vibration of LWAC promotes segregation, in which denser aggregates settle to the bottom while the lower-density LWA tends to float, which affects its compressive strength and durability [6]. Over-vibration has been found to reduce the compressive strength by up to 67% depending on the density and the LWA used [7].

There are several factors that affect the segregation of concrete, for example, the components of the material, the rheology of the mortar, the sizes of the particles, the density of the produced concrete, and the compaction procedure, which induces anisotropy in the casting direction and has a negative impact on the properties of the concrete [8,9]. There are several different methods of quantifying the segregation degree of concrete, most of which are based on standard density measurements, ultrasonic velocity measurements or image analysis [6,10–14]. In particular, the ultrasonic pulse velocity method is a non-destructive approach that has been successfully used to evaluate the quality of concrete. It can also be used to detect internal cracking and other defects as well as changes in the concrete.

The use of machine learning techniques has become an important area of research for predictive analysis in engineering (see, e.g., [6,15–17] and the references therein). The enormous variety of formulae used to create LWAC with specific qualities and the continuously increasing complexity of cementitious systems remain challenging problems. The ability of machine learning to tackle complex tasks autonomously means that it has transformative potential for research into concrete [18]. In the context of building and construction materials, several works have reported the use of artificial neural networks (ANNs) to predict the compressive strength of concrete. For example, in [6], a multilayer perceptron (MLP) network was employed to estimate the influence of the segregation in concrete on the compressive strength of LWAC using ultrasonic pulse velocity. The Levenberg–Marquardt backpropagation algorithm was used to train the network, and the best prediction model for the compressive strength of LWAC gave a determination coefficient of about 0.825 and a root mean square error (RMSE) of about 3.745. In [19], the behaviour of multiple linear regression (MLR) and regression tree (RT) models were compared with the ANN models proposed in [6], using the same dataset, and the authors showed that MLR models were not competitive in terms of explaining the compressive strength of LWAC, with a poor determination coefficient of about 0.77 and an RMSE of about 4.332. In addition, CHAID (Chi-squared automatic interaction detector) [20], exhaustive CHAID [21] and CRT (classification and regression trees) [22] were analysed. The best RT model was obtained with the CHAID algorithm, which yielded performance similar to that of the ANN models proposed in [6], with a determination coefficient of about 0.82 and an RMSE of about 3.808.

In other studies, MLR and ANN models have also been compared in terms of predicting some of the properties of concrete. For example, in [23], ANNs and MLR were used to predict the compressive strength of concrete based on two predictor variables (the ultrasonic pulse velocity and the weight of concrete specimens), whereas in [24], categorised linear regression, MLR and ANN were employed to predict the compressive strength of structural LWAC. The results of both of these works indicated the higher accuracy of ANN models. In [25], MLR and ANN models were used to predict slump as well as 7-day and 28-day compressive strengths based on data obtained from plants in India using ready-mix concrete. The ANN models outperformed the MLR models, giving more accurate predictions for both slump and compressive strength. In [26], the use of ANNs to predict the compressive strength of recycled brick aggregate concrete was explored. The components of the concrete mixture formed the input data parameters (cement, water–cement ratio, crushed tile ratio, crushed brick ratio and natural aggregate ratio), and the results showed

that the type of clay aggregate used in the study (brick or tile) could be neglected. In [27], ANN, nonlinear regression (NLR) and RT models were compared to predict the 28-day compressive strength of recycled aggregate concrete. The ANN models outperformed both the NLR and RT models with nine predictor variables (cubic metre proportions of cement, natural fine aggregate, recycled fine aggregate, natural coarse aggregate—10 mm, natural coarse aggregate—20 mm, recycled aggregate—10 mm, recycled aggregate—20 mm, admixture and water). RT methods have also previously been applied to predict the elastic modulus of recycled aggregate concrete [28] and to address other problems in the area of civil engineering, such as the modelling of damage in reinforced concrete buildings [29]. In [30], a backpropagation ANN model was proposed to predict the bond strength for fibre-reinforced plastic (FRP) reinforced concrete at high temperatures. This ANN was tested by using 151 data points, and its precision was found to be slightly lower than that of the current mathematical models [31–33]. However, it had higher generality in predicting the bond strength when applied to inexpressible problems involving multiple influence parameters.

In [34], two-level and hybrid ensembles of RT were used to predict the compressive strength of high-performance concrete and were found to outperform RT models. Recently, in [35], support vector regression (SVR), MLP, gradient boosting regressors (GBRs) and extreme GBR (or XGBoost) were also used to predict the compressive strength of high-performance concrete. The results were better than those reported in other studies [36–40], in which several machine learning techniques were proposed for the same problem. As reported in [34,35], tree-based ensemble methods have strong potential in terms of predicting the compressive strength of high-performance concrete (see also [41]).

In [42], a comparative study of machine learning methods was conducted in the context of predicting the strength of concrete with blast furnace slag. *K*-nearest neighbours (KNN), SVR and random forest (RF) achieved high levels of prediction accuracy. In [43], MLP, RF and KNN algorithms were used to predict the compressive strength of concrete mixed with ground granulated blast furnace slag and fly ash. The results showed similar performance for the RF and MLP regression models, which outperformed the KNN model. Moreover, RF was the most stable and was the least influenced by the data splitting process.

Recently, in [44,45], various machine learning algorithms were proposed to predict the compressive strength of lightweight concrete. In [44], Gaussian progress regression (GPR), SVR, and ensemble learning were applied to a dataset consisting of 120 data points. The predictor variables were cement content, water content, fine aggregate, normal weight coarse aggregate, lightweight coarse aggregate, and water-cement ratio. The dataset was partitioned into 70% for training and 30% for testing. To obtain the best fitted models, 10-fold cross-validation was applied to the training dataset. The optimised GPR and SVR models obtained the best accuracy, with training correlation coefficients of 0.9933 and 0.9947, respectively, and testing correlation coefficients of 0.8915 and 0.8882. The optimised SVR gave an RMSE of 3.2988 for the training dataset and 4.6111 for the test dataset, while the optimised GPR achieved values for the RMSE of 1.7982 and 4.4002, respectively. The study performed in [45] utilised 420 data points and applied the SVR, ANN, RT, GPR, and XGBoost algorithms to predict the compressive strength of LWAC. The predictor variables included cement content, sand content, water–cement ratio, quantity of LWA, normal aggregate content, density of LWA, water absorption of LWA, proportion of superplasticiser, curing time, fly ash content, and type of LWA. To obtain the best models, the parameters involved in the algorithms and the percentages of the data used for training, validation and testing were varied. The best results were achieved with the GPR, RT and SVR algorithms, although the GPR models outperformed all of the other models, with a training RMSE of 1.34 and a testing RMSE of 5.06. The values obtained for the mean absolute error (MAE) were 0.69 and 3.01, respectively.

In this work, we focus on the use of machine learning techniques to predict the compressive strength of LWAC with expanded clay using ultrasonic pulse velocity. The LWAC is compacted for different vibration times in order to segregate the produced concrete.

The detailed experimental procedure for obtaining the dataset used in this work is explained in [6] and summarised in Section 2.2. KNN, ANN, SVR, RF and several ensemble algorithms are employed to estimate the compressive strength of LWAC using ultrasonic pulse velocity. In addition, a weighted ensemble (WAE) algorithm that combines RF, SVR and XGBoost is proposed. The input variables include the fixed density of LWAC, the particle density of LWA, the concrete laying time, the vibration time, the experimental dry density, the *P*-wave velocity and the segregation index based on *P*-wave velocities.

Section 2 describes the materials and methods used in this work. After providing an overview of machine learning techniques and the methodology used to evaluate the models, Section 2 also explains the problem addressed in this paper. Section 3 analyses the behaviour of the proposed machine learning methods and discusses the numerical results obtained in this work. For each machine learning technique, an exhaustive analysis of its behaviour is presented based on the values of the parameters and hyperparameters involved. Once the best models have been obtained for each technique, they are compared to find the best machine learning techniques to predict the compressive strength of LWAC. Numerical experiments show that gradient-boosted tree models and the proposed WAE models give better performance than the models obtained using other machine learning techniques. Finally, some conclusions are presented in Section 4.

2. Materials and Methods

Machine learning is a field of artificial intelligence that focuses on the development of algorithms which, based on the analysis of datasets, can learn from patterns in these data and apply what they have learned to decision making. Using the dataset described in [6], several machine learning algorithms are proposed and analysed in terms of predicting the compressive strength of LWAC, including KNN, ANNs, SVR, RF and other tree-based ensemble methods. Our experimental results are also compared with those reported in previous works [6,19], in which MLR, RT and ANN models were applied to the same dataset. Section 2.1 explains the machine learning techniques involved in this research, the validation process used and the criteria applied for model selection. Section 2.2 describes the dataset used in the experimental study and summarises the experimental results obtained in previous works for this dataset.

2.1. Machine Learning Methods

2.1.1. Multiple Linear Regression

An MLR model has the form:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_d X_d + \epsilon, \tag{1}$$

where Y is the dependent variable, $X_1, X_2, \ldots, X_d$ are the independent or predictor variables, $\beta_0, \beta_1, \beta_2, \ldots, \beta_d$ are the regression coefficients, and ϵ is the random component of the model (that is, the unpredictable part of the dependent variable).

If we assume that the dataset has n instances, then the model defined in Equation (1) can be written as follows:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_d x_{id} + \epsilon_i,\ i = 1, 2, \ldots, n.$$

To ensure the validity of an MLR model, the relationship between the dependent variable and each of the predictor variables must be linear, and the residuals ϵ_i, $i = 1, 2, \ldots, n$, must be normally distributed with $\mathrm{E}(\epsilon_i) = 0$ and $\mathrm{Var}(\epsilon_i) = \sigma^2$. It is also assumed that there is no multicollinearity between predictor variables.

2.1.2. *K*-Nearest Neighbours

KNN algorithms are widely used for both classification and regression problems. When applied to regression problems, this supervised machine learning method attempts to predict the values of a quantitative variable Y from the values obtained for the pre-

dictor variables $X_1, X_2, \ldots X_d$, based on a similarity metric. That is, given a training set $\{(x^{(1)}, y_1), (x^{(2)}, y_2), \ldots, (x^{(m)}, y_m)\}$ with $x^{(i)} = (x_{i1}, \ldots, x_{id})$, $i = 1, 2, \ldots, m$, the K shortest distances between the data to be evaluated and the training data are obtained. The predicted value for a test instance x, denoted as $\hat{y}(x)$, is then obtained as the average of the outputs of its K nearest neighbours (denoted as $y_i(x)$, $i = 1, 2, \ldots, K$). That is, $\hat{y}(x) = \frac{1}{K} \sum_{i=1}^{K} y_i(x)$ (see Figure 1). The performance of this method depends on both the value selected for K and the type of metric chosen for the KNN algorithm [46,47]. Two well-known metrics used in this method are the Euclidean and Manhattan distances. Although the metric may affect the performance of this method, there seems to be little consensus on which distance is most generally applicable and in which cases [48].

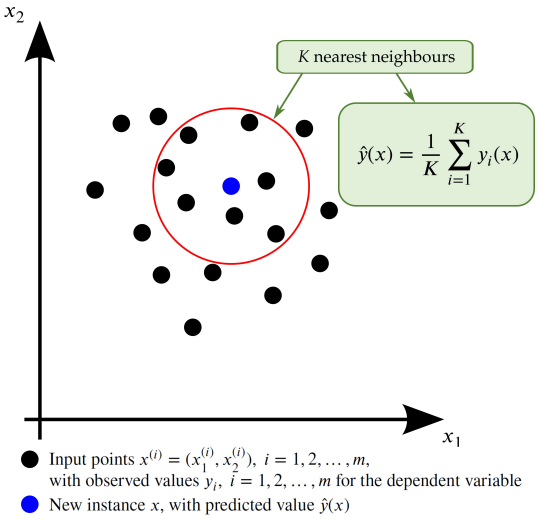


Figure 1. KNN algorithm for $d = 2$ predictor or dependent variables and $K = 7$.

2.1.3. Artificial Neural Networks

ANNs are a set of useful machine learning techniques based on biological neural networks and are used to model complex relationships between inputs and outputs or to find patterns. The simplest processing element of a neural network is the neuron. Each neuron i may have multiple inputs, $x_1, x_2, \ldots, x_d$, and synaptic weights, $w_{i1}, w_{i2}, \ldots, w_{id}$ from which it produces a single output. More specifically, an input or propagation function g_i combines these inputs with their synaptic weights, and an activation function f_i is then subsequently applied to the obtained result to generate the corresponding output. Figure 2 graphically illustrates the model of a neuron i, in which the input function is defined as $g_i(x_1, x_2, \ldots, x_d, w_{i1}, w_{i2}, \ldots, w_{id}) = \sum_{j=1}^{d} w_{ij} x_j + \theta_i$, and θ_i is a bias value. This value is treated as a weight whose input is one. The output value of neuron i can therefore be expressed as $y_i = f_i(\sum_{j=1}^{d} w_{ij} x_j + \theta_i)$, where f_i is some activation function.

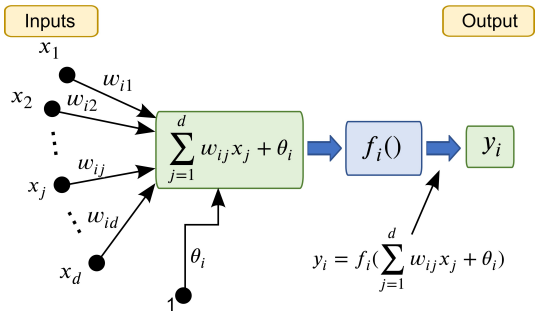


Figure 2. Diagram of an artificial neuron.

Although a single neuron can perform some simple information processing functions, the power of ANNs arises from the connections between many of these simple and robust

units. Two typical network architectures, which differ in the types of connections between neurons, are the feedforward and recurrent networks. In a feedforward ANN, there are no cycles; that is, there is no closed path from any neuron to itself, meaning that the flow of information has a single direction. In a recurrent ANN, however, cycles are present, meaning that information can flow in any direction between the different layers, including the input layer and the output layer.

An MLP network is a type of fully connected feedforward ANN in which backpropagation is used for training the network [49]. There are three types of layers in this neural network: the input layer, the hidden layers, and the output layer. Neurons in the input layer receive inputs and propagate them to the next layer. The hidden layers form the intermediate layers, while the output layer transfers the information from the network to the outside and gives the response for each input pattern. Figure 3 shows an example of an MLP network with a single hidden layer, where $\{x_1, x_2, \ldots, x_d\}$ is the set of inputs accessing the network, w_{ij} denotes the synaptic weight between input neuron j and hidden neuron i, and $\tilde{w}_{ki}$ denotes the synaptic weight between hidden neuron i and output neuron k. As can be seen from this figure, the activation propagates in the network through the weights from the input layer to the intermediate layer. In this hidden layer, a certain activation function is applied to the inputs that reach it. The activation then propagates through the synaptic weights to the output layer. To train the network, two sets of weights must be updated: those between the input layer and the hidden layer, and those between the hidden layer and the output layer. Various activation functions can be used in an MLP network depending on the characteristics of the problem. Some of the most commonly used for regression problems are the logistic sigmoid function, the hyperbolic tangent function (tanh), and the rectified linear unit (ReLU) function [50].

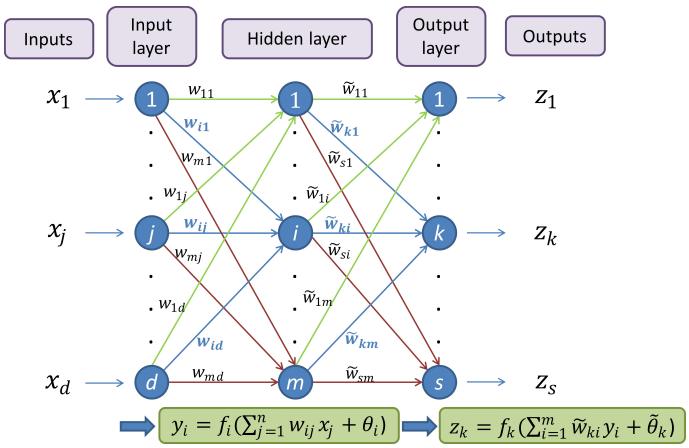


Figure 3. Example of MLP network.

To create the predictive model, a supervised and iterative learning process is followed, starting with an initial configuration of synaptic weights. These weights are adjusted iteratively according to the inputs of the training set, and a loss function is used to compare the output with the data to be predicted. One widely used loss function for regression problems is the mean squared error (MSE). The learning algorithm attempts to find the weights of the neural network that minimise this function. Several optimisation algorithms can be used for this purpose, for example, stochastic gradient descent (SGD), Adam and limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) [51]. Adam [52] is a stochastic gradient-based optimiser that typically performs better on relatively large datasets. L-BFGS [53] is an optimiser from the family of quasi-Newton methods that tends to perform better for small datasets. Each optimiser has several associated parameters that should be suitably chosen to improve training and reduce the cost. These parameters, called hyperparameters in the context of neural networks, are configured before network training is carried out but do not form part of its architecture.

2.1.4. Regression Trees and Tree-Based Ensemble Methods

An RT is a variant of the decision tree model [54] in which the target variable is continuous. Using a set of decision rules, the interdependence between the predictor variables and the target variable is graphically represented in the form of a tree structure. This structure is defined by the topology of the tree (configuration of nodes and arcs) and the splitting rules applied to build the tree. Given a training set $\{(x^{(1)}, y_1), (x^{(2)}, y_2), \ldots, (x^{(m)}, y_m)\}$ with $x^{(i)} = (x_{i1}, \ldots, x_{id})$, $i = 1, 2, \ldots, m$, and with the aim of predicting the values of the quantitative variable Y from the values obtained for the predictor variables $X_1, X_2, \ldots X_d$, a recursive process is used to build the tree, starting from the root node representing the entire current dataset. The dataset is then partitioned into smaller and smaller subsets based on the input variables considered in an attempt to obtain similar or homogeneous behaviour within nodes with respect to the target variable Y. In an RT, the space of all joint predictor variable values is partitioned into disjoint regions R_j, $j = 1, 2, \ldots, J$, which are represented by the terminal nodes of the tree. A constant value γ_j is assigned to each region, and the predictive rule is obtained as $x \in R_j \rightarrow f(x) = \gamma_j$. Hence, a tree can be expressed as $T(x; \Theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j)$ with parameters $\Theta = \{R_j, \gamma_j\}_1^J$, where $I(\cdot)$ denotes the indicator function of a set. The parameter values are found with the aim of minimising some specified loss function L, that is, $\hat{\Theta} = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, \gamma_j)$ [55].

In this context, some well-known RT algorithms are CHAID, exhaustive CHAID, and CRT. We note that when they are used for prediction in regression problems, these algorithms use the mean of the corresponding nodes. In addition, CHAID and exhaustive CHAID use the F-test as a splitting criterion [56]. The CRT algorithm builds binary trees to maximise within-node homogeneity. The node impurity is a measure of the homogeneity between the values of the node, and in regression problems, the impurity is typically measured using the least-squared deviation [19].

However, overfitting is a common problem in decision trees, and very different trees can be obtained with small changes in the training dataset. To overcome this limitation, RF can be used [57]. RF is an ensemble machine learning method that combines several decision trees based on various random subsamples (see Figure 4). The construction of the RF model is usually done using bagging (bootstrap aggregation). The bagging technique uses several training datasets obtained from the full dataset by random sampling with replacement. Each training dataset is then used to build a tree model. That is, in RF, each tree is trained independently and at the same time. An ensemble of these strong learners, using for example the average of all of them, produces a more robust model than a single decision tree [58]. The important hyperparameters in RF include the number of trees used to build the model, the function chosen to measure the quality of each split (MSE, MAE, etc.), and several useful parameters for controlling the growth of the trees. Increasing the number of trees when building an RF model does not cause overfitting.

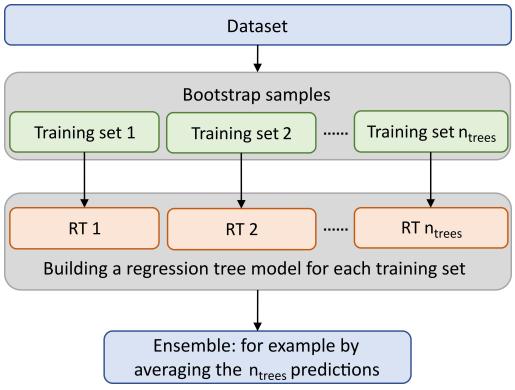


Figure 4. Flowchart of the RF algorithm.

In contrast, boosting is an ensemble iterative technique in which each new tree is built in an attempt to minimise the errors of previous trees, and thus each tree depends on the previous one. This technique minimises the loss function by adding, at each step, a new tree that best reduces the loss function. A well-known method of this type is GBR [59,60]. Algorithm 1 describes this technique. Unlike RF, in which all trees are constructed independently, each using a subset of the training dataset, GBR uses gradient boosting as an ensemble technique [61].

Algorithm 1 GBR algorithm

Let M be the number of boosting iterations.
Given a learning rate ν, $0 < \nu \leq 1$.
Initialise $f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{m} L(y_i, \gamma)$.
for $k = 1 \to M$ **do**
  for $i = 1 \to m$ **do**
    Compute $r_{ik} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{k-1}}$.
  end for
  Fit a regression tree to the targets r_{ik} giving terminal regions R_{jk}, $j = 1, 2, \ldots, J_k$.
  for $j = 1 \to J_k$ **do**
    Compute $\gamma_{jk} = \arg\min_{\gamma} \sum_{x_i \in R_{jk}} L(y_i, f_{k-1}(x_i) + \gamma)$.
  end for
  Update $f_k(x) = f_{k-1}(x) + \nu \sum_{j=1}^{J_k} \gamma_{jk} I(x \in R_{jk})$.
end for
return $\hat{f}(x) = f_M(x)$.

In this approach, an additive model is built such that at each stage, an RT is fitted to the negative gradient of the given loss function. Note that at each boosting iteration k, a tree is added that modifies the overall model. The added trees are weak learners in the sense that their performance is slightly better than random chance. An important parameter associated with gradient-based optimisers is the learning rate, which controls the contribution of each tree to the final outcome. If the learning rate is low, more boosting stages (i.e., more trees) are needed in the training process. In general, models that learn slowly perform better. However, the number of boosting iterations must be carefully selected due to the high risk of overfitting [59]. In addition, the extreme GBR (or XGBoost) method may be used to reduce overfitting. This is an extension of the gradient boosting method that employs advanced regularisation techniques such as Lasso regularisation (based on the L1 norm), Ridge regularisation (based on the L2 norm), and dropout techniques. The Lasso and Ridge regularisation methods introduce an additional term into the loss function to penalise large weights [62]. The dropout technique [63] is an adaptive regularisation method in which a random percentage of trees is ignored in the training phase so that no tree manages to memorise part of the inputs [62].

2.1.5. Support Vector Regression

SVR is an extension of the support vector machine classification algorithm [64] and is used for regression problems [65] such as the one considered here. To obtain the estimated continuous-valued multivariate function with this technique, an ϵ-insensitive region around the function, called the ϵ-tube, is considered. The optimisation problem then involves finding the tube that best approximates it. The estimation function is obtained by mapping the input samples onto a higher-dimensional feature space using a nonlinear mapping ϕ and learning a linear regressor in the feature space [66]. That is, given the training vectors $x^{(i)} \in \mathbb{R}^d$, $i = 1, 2, \ldots, m$, and a vector $y \in \mathbb{R}^m$, it is assumed that the regression estimating

function $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ is defined by $f(x) = \omega^t \phi(x) + b$, where ω is a vector in the feature space and b is a scalar threshold. Then, to determine the values of ω and b, SVR solves the following primal problem [64,67,68]:

$$\begin{aligned} \min_{\omega,b,\xi,\xi^*} \frac{1}{2}\omega^t\omega + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \\ \text{subject to } y_i - \omega^t\phi(x^{(i)}) - b \leq \epsilon + \xi_i, \\ \omega^t\phi(x^{(i)}) + b - y_i \leq \epsilon + \xi_i^*, \\ \xi_i, \xi_i^* \geq 0,\ i = 1, 2, \ldots, m, \end{aligned}$$

where $C, \epsilon \in \mathbb{R}^+$ are input parameters and ξ, ξ^* are nonnegative vectors of slack variables. This optimisation problem can be solved in its dual form using the Lagrange multipliers method, as follows:

$$\begin{aligned} \min_{\alpha,\alpha^*} \frac{1}{2}(\alpha - \alpha^*)^t P(\alpha - \alpha^*) + \epsilon e^t(\alpha + \alpha^*) - y^t(\alpha - \alpha^*) \\ \text{subject to } e^t(\alpha - \alpha^*) = 0, \\ 0 \leq \alpha_i, \alpha_i^* \leq C,\ i = 1, 2, \ldots, m, \end{aligned}$$

where $\alpha, \alpha^* \in \mathbb{R}^m$ are the Lagrange multipliers, e is a vector of all ones and P is an $m \times m$ positive semidefinite matrix, with $P_{ij} = \phi(x^{(i)})^t\phi(x^{(j)})$. Then, taking $K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^t\phi(x^{(j)})$, where K is a kernel function, the prediction of an input vector $x \in \mathbb{R}^d$ is obtained as $f(x) = \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)K(x^{(i)}, x) + b$ (see Figure 5).

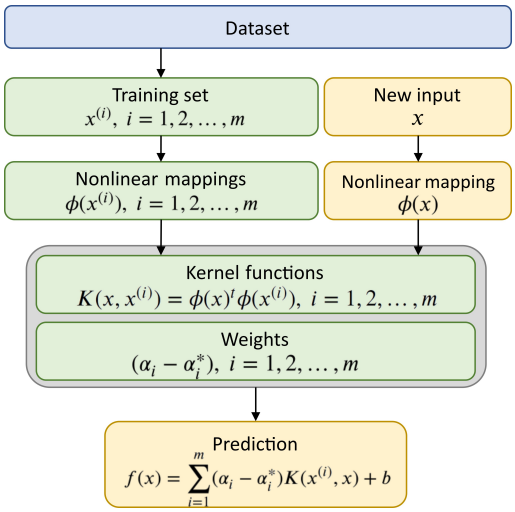


Figure 5. Flowchart of the SVR algorithm.

To rewrite and solve the dual problem, various kernel functions can be used depending on the characteristics of the dataset, such as polynomial functions ($K(x^{(i)}, x^{(j)}) = (\gamma\langle x^{(i)}, x^{(j)}\rangle + \tau)^\delta$), sigmoid functions ($K(x^{(i)}, x^{(j)}) = \tanh(\gamma\langle x^{(i)}, x^{(j)}\rangle + \tau)$) or radial basis function (RBF) kernels ($K(x^{(i)}, x^{(j)}) = e^{-\gamma\|x^{(i)} - x^{(j)}\|^2}$). In the training phase, different parameters (C, ϵ, γ, δ, τ) must be optimised depending on the kernel function, although C and ϵ are common to all SVR kernels.

We note that SVR solves a convex optimisation problem in which the optimal solution is found analytically rather than heuristically, such that for a predetermined kernel and a given training dataset, the training phase returns a uniquely defined model.

2.1.6. Validation Process and Criteria Used for Model Selection

Validation of the model is a critical stage in machine learning. There are several different validation techniques (e.g., holdout, random subsampling, resubstitution, k-fold cross-validation, and leave-one-out cross-validation) [69]. In this work, we use random

subsampling and repeated k-fold cross-validation. Random subsampling or Monte Carlo cross-validation is based on random splitting of the dataset into several training and test sets. A prediction model is obtained for each training dataset, and is evaluated on the corresponding test dataset. To analyse the performance of a model, the average values of the goodness-of-fit measures are considered (see Figure 6a). In k-fold cross-validation, the dataset is randomly partitioned into k subsamples of equal size. At each iteration, one of these k subsamples is used as a test set, and the remaining $k-1$ subsamples as a training set. The performance of this approach is obtained based on the average values of the goodness-of-fit measures at each iteration (see Figure 6b). In repeated k-fold cross-validation, this process is repeated several times with different random data partitions, and the mean performance across all repeats is considered.

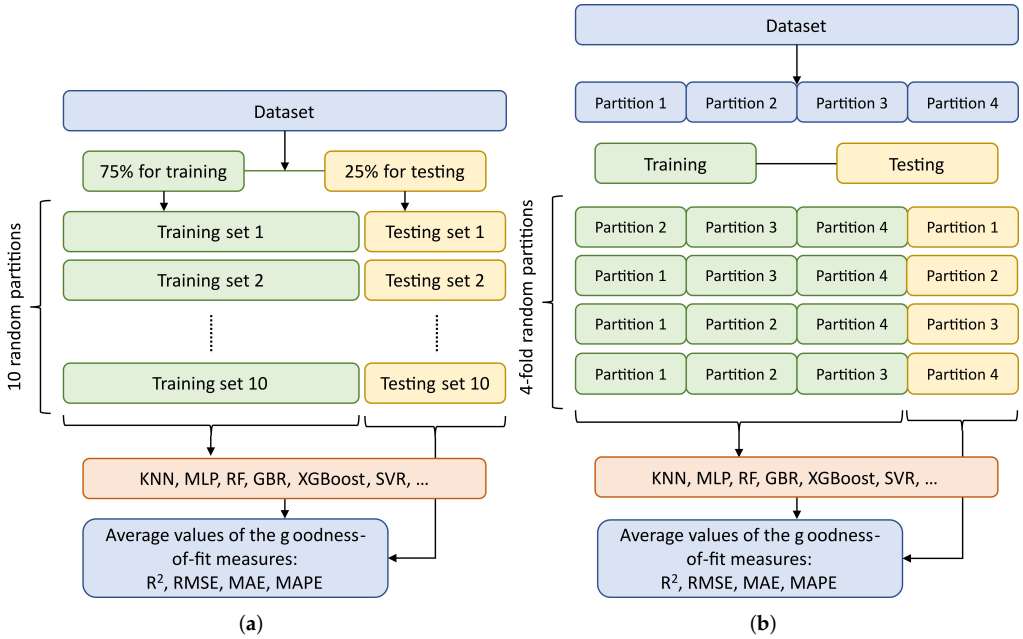


Figure 6. Random subsampling and k-fold cross-validation procedures. (**a**) A 10 random subsampling procedure by randomly partitioning the sample into 75% for training and 25% for testing; (**b**) k-fold cross-validation procedure, example with $k = 4$.

To assess the performance of the models presented in this study, the following goodness-of-fit measures were considered (where y_i, $i = 1, 2, \ldots, n$, denote the observed values; $\bar{y}$ is the mean of the observed values; $\hat{y}_i$, $i = 1, 2, \ldots, n$, are the predicted values; and d is the number of predictive variables in the model):

- Relative error, $\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$.
- Determination coefficient, $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$.
- Mean squared error, $\text{MSE} = \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n}$.
- Root mean square error, $\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n}}$.
- Mean absolute error, $\text{MAE} = \frac{\sum_{i=1}^{n}|y_i-\hat{y}_i|}{n}$.
- Mean absolute percentage error, $\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i-\hat{y}_i}{y_i}\right|$.

The determination coefficient is one of the most frequently used measures of the degree of linear correlation between two variables. This measure can be interpreted as the proportion of variability explained by the model. Note that the determination coefficient in a linear least-squares regression with an intercept is equivalent to the square of the correlation coefficient and, therefore, takes values between zero and one, where a value of one indicates perfect correlation. However, outside of this context, the determination

coefficient may be negative. The RMSE, MAE and MAPE are also very suitable measures for comparing the performance of the models, where lower values indicate a better fit of the model. The MAPE and MAE are robust to the effects of outliers, due to the use of absolute values in the formulae, while the RMSE is more sensitive. Nevertheless, there are some circumstances in which the use of the RMSE is more beneficial (for example, when the errors are normally distributed), meaning that it would be hard to argue that one measure is better than another. Hence, a combination of these metrics is used to analyse the performance of the models.

One of the main concerns in machine learning is overfitting or underfitting the models. Underfitting occurs when the model is not able to represent the training data or new inputs that are not included in the training dataset. Although in some cases this is due to insufficient training, the most common causes of underfitting are poor choices of the training dataset and learning model. On the other hand, overfitting is usually due to excessive training. If the model is overtrained, it will be able to model the training data almost perfectly, but with new datasets, it may not achieve high accuracy. These issues, therefore, need to be analysed to obtain the best models.

On the other hand, to compare the models, an analytic hierarchy process based on discrete ranking analysis [45,70] was also performed. In this case, the models were rated based on the goodness-of-fit measures obtained for the training and test datasets (see Figure 7a). The models were then ordered from lowest to highest performance for each of these measures. The rating assigned to the worst model was one, while the rating assigned to the best model was equal to the number of models considered in the study. The rating assigned to each model for a goodness-of-fit measure was then computed as the sum of the two ratings. The overall rating of a model was computed by adding the individual ratings obtained for the considered goodness-of-fit measures. To compute this overall rating, we used the determination coefficient, RMSE, MAE and MAPE measures. In addition, to consider the magnitude of the performance differences, a continuous ranking was proposed (see Figure 7b). This ranking uses normalised values (max-normalisation) of the relative error ($1 - R^2$), RMSE, MAE and MAPE rather than discrete values. The sum of these values for the training and test datasets gives a rating for each model, which also shows the differences between the performance of one model and the others. Lower values of this continuous rating indicate higher performance.

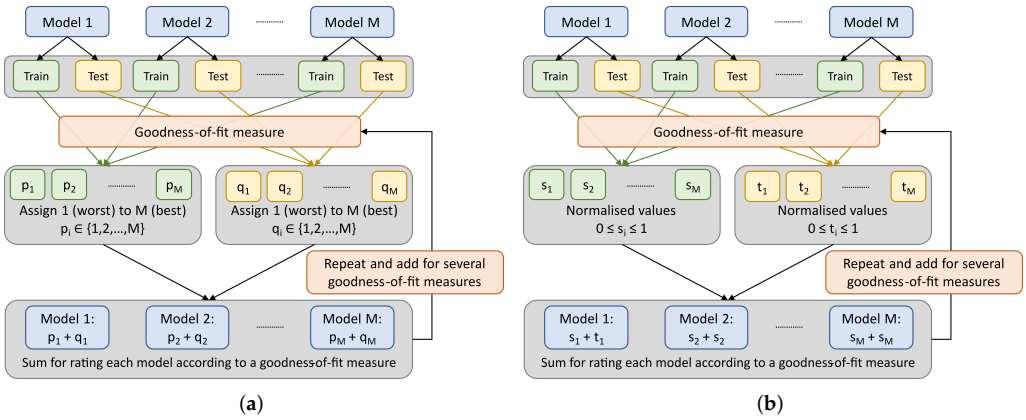


Figure 7. Ranking a set of models. (**a**) Discrete ranking; (**b**) continuous ranking.

2.2. Dataset and Related Results

The dataset consisted of 640 samples from four different types of LWAC that were intentionally segregated by compacting them for different vibration times, following the experimental procedure explained in [6] (see Table 1). To manufacture the samples, each of the four types of concrete was mixed in a vertical shaft, and cylindrical moulds (with diameter 150 mm and height 300 mm) were filled and compacted using a needle vibrator (for 0, 10, 20, 40 and 80 s). The laying time was also taken into account in order to simulate

the laying of LWAC under real conditions, as fresh concrete loses its workability over time. The mixture proportions of each type of LWAC were computed according to the Fanjul method [71]. A total of 160 cylinders were manufactured. The type of LWAC depended on both the type of LWA used to produce the concrete and the target density of the produced concrete (1700 or 1900 kg/m^3). Two types of LWA were considered, consisting of expanded clay (from the company Saint-Gobain) with different particle densities, the first with a density of 482 kg/m^3 and a granulometric fraction with sizes 6/10, and the second with a density of 1019 kg/m^3 and fraction 4/10. The lighter LWA has a high-porosity structure (36.6% of water absorption at 24 h) and is typically used for ultra-lightweight concrete applications, whereas the denser LWA (12.20% of water absorption at 24 h) is mainly used for structural concrete applications. Two cores of diameter 50 mm were extracted from each LWAC cylinder produced in this way and were cut into four 70 mm lengths. The variables considered in this problem were the theoretical density of the concrete according to the Fanjul method [71], the particle density of the LWA, the laying time of the concrete, the vibration time, the dry density of the specimen obtained experimentally after 28 days, the *P*-wave velocity and the segregation index. The segregation index used in this work was based on the *P*-wave velocities and was proposed in [6] as an alternative to the segregation index put forward by Ke [12] for the case where several cores from a concrete specimen are considered rather than one complete specimen. To obtain the segregation index based on the *P*-wave velocity, the longitudinal velocity (V_p) of each core was determined using 250 kHz transducers. The mean value of the *P*-wave velocity of an LWAC was then computed based on the four *P*-wave velocities of the cores (V_{pm}), considering the position of the core in the original specimen (V_{pSi}). The segregation index was computed for each core as the ratio between the *P*-wave velocity of the core (V_{pSi}) and the mean *P*-wave velocity of the four cores of the corresponding LWAC cylinder (V_{pm}) [6]. To determine the influence of segregation, the compressive strength of each core sample was obtained using a 200 kN machine with a loading rate of 0.25 MPa/s. A preliminary statistical study showed significant differences in compressive strength depending on the type of LWAC. As expected, the higher the LWA particle density or LWAC density, the greater the compressive strength [19]. Table 2 summarises the statistical characteristics of the variables involved in this problem: minimum (min), maximum (max), mean, median, first quartile (Q_1) and third quartile (Q_3).

Table 1. Characteristics of types of LWAC. "Adapted with permission from Ref. [19]. Copyright 2019, Elsevier".

LWAC Type	LWA Particle Density (kg/m^3)	LWAC Fixed Density (kg/m^3)
1	482	1700
2	482	1900
3	1019	1700
4	1019	1900

Table 2. Variables of the experimental dataset. "Adapted with permission from Ref. [6]. Copyright 2018, Elsevier".

Variable Description	Min	Max	Mean	Median	Q_1	Q_3
LWAC fixed density (kg/m^3)	1700	1900	1800	1800	1700	1900
LWA particle density (kg/m^3)	482	1019	750.5	750.5	482	1019
Concrete laying time (min)	15	90	48.75	45.00	18.75	82.50
Vibration time (s)	0	80	30	20	10	40
Experimental dry density (kg/m^3)	1069.80	2486.84	1673.35	1677.15	1533.35	1810.84
P-wave velocity (m/s)	3044.25	5253.73	3778.89	3718.49	3520.48	3945.65
Segregation index	0.845	1.136	1	0.999	0.978	1.021
Compressive strength (MPa)	2.99	50.72	21.55	20.25	14.37	28.76

We are interested in predicting the compressive strength of LWAC by applying machine learning to this dataset. This problem has been considered in previous works in which MLR, ANNs and three RT algorithms (CHAID, exhaustive CHAID, and CRT) were used to predict the compressive strength of LWAC [6,19]. Tables 3 and 4 summarise the best models obtained in these works. However, there are no experiments in the literature that have analysed the performance of other machine learning techniques for predicting the compressive strength of concrete with this dataset.

Table 3. Statistical measures for the best MLR and RT models in [19]. "Adapted with permission from Ref. [19]. Copyright 2019, Elsevier".

Technique	Model	Measures	Estimate
MLR	Without validation	R^2	0.767
		MAE	3.394
		MAPE	19.04
		RMSE	4.327
	Best model with validation	R^2	0.766
		MAE	3.396
		MAPE	18.86
		RMSE	4.332
CHAID	Without validation	R^2	0.829
		MAE	2.829
		MAPE	15.49
		RMSE	3.705
	Best model with validation	R^2	0.820
		MAE	2.928
		MAPE	16.22
		RMSE	3.808

Table 4. Statistical measures for the best ANN model in [6]. "Adapted with permission from Ref. [6]. Copyright 2018, Elsevier".

Measures	Estimate
R^2	0.825
MAE	2.897
MAPE	15.85
RMSE	3.745

3. Results and Discussion

To implement and construct our machine learning models, we used Python 3.9, version 1.1.2 of the Scikit-learn API [72] and version 1.6.2 of the XGBoost (extreme Gradient Boosting) library [62]. Other Python libraries such as NumPy 1.21.0, SciPy 1.7.0, and pandas 1.3.0 were also needed. The experiments were carried out on a Windows 10 64-bit PC with an Intel Core i7-1065G7 CPU at 1.30 GHz and 16 GB of RAM. To evaluate each model using Monte Carlo cross-validation, the sample was split into two datasets: the training set, which was used to estimate the new model, and a test set, which was used to evaluate the predictive ability of the model. The validation process was repeated 10 times, by randomly partitioning the sample into 75% for training and 25% for testing. Several experiments were also performed with repeated k-fold cross-validation, using different values of k ($k = 4$, $k = 10$). The k-fold cross-validation procedure was also repeated 10 times. To enable a comparison and ensure the reproducibility of results, the same 10 seeds were used to perform the partitions in both cross-validation procedures (see also Section 2.1.6).

At the preprocessing stage, normalisation was applied to scale the dataset. This is often useful to improve the accuracy of most machine learning algorithms, especially

when there are variables with different units. In the same way as in [6], we used min-max normalisation in this work to scale the data to the range $[0, 1]$.

For each machine learning technique, an exhaustive analysis of its behaviour is presented below, including the values chosen for the parameters and hyperparameters involved in each method. Based on these analyses, the best models are selected in an attempt to avoid both overfitting and underfitting.

3.1. KNN Algorithms

In this section, we analyse the behaviour of the KNN algorithms in terms of predicting the compressive strength of LWAC. Two similarity metrics were used, the Euclidean and Manhattan distances, and the neighbourhood size K was varied from 1 to 50. Figure 8 illustrates the influence of the type of distance used in the KNN algorithm. The mean values of the RMSE, MAE and MAPE for the test datasets of the 10 random models are shown in Figure 8a–c, respectively. The best results were achieved with the Manhattan distance and values of K of between five and seven. For these values of K, Figure 9 displays the RMSE of the 10 random runs of the validation process. For the training dataset, the best results were obtained with $K = 5$, with a mean RMSE of 3.8173, a mean MAE of 2.8956 and a mean MAPE of 15.3229%. For the test data, the best mean RMSE and MAE were obtained with $K = 7$ (4.3365 and 3.3134, respectively) and the best mean MAPE with $K = 5$ (see Table 5). Table 6 displays several goodness-of-fit measures for the model with these values of K, obtained both without validation and after the validation process. The same results were obtained with both Monte Carlo and repeated k-fold cross-validation. The best model achieved a coefficient of determination of 0.812 and an RMSE of 3.8863. The best model without validation had a coefficient of determination of 0.8365 and an RMSE of 3.6246.

A comparison of these models with those presented in [19] (see Table 3) and the ANN models in [6] (see Table 4) shows that the KNN algorithms (using the Manhattan distance) outperformed the MLR models (RMSE = 4.332) in predicting the compressive strength of LWAC. However, KNN showed slightly lower performance than the best tree model in [19] (RMSE = 3.808) and the best ANN model in [6] (RMSE = 3.745).

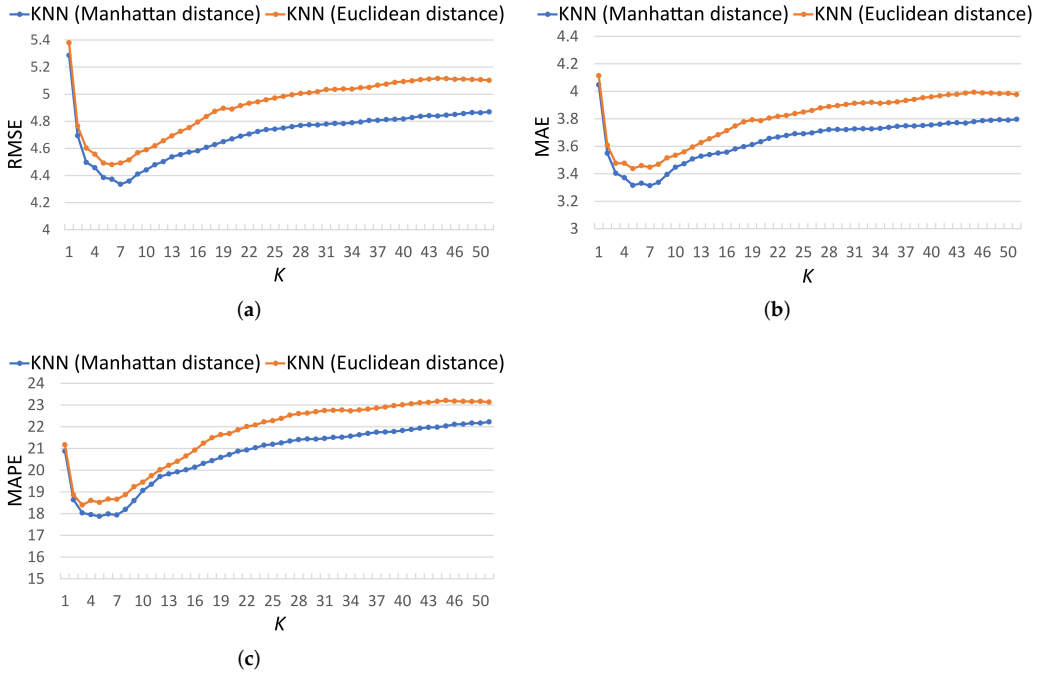


Figure 8. Comparison of KNN models, $1 \leq K \leq 50$; mean values of statistical measures on the test data for the 10 random models (75:25 ratio). (**a**) RMSE; (**b**) MAE; (**c**) MAPE.

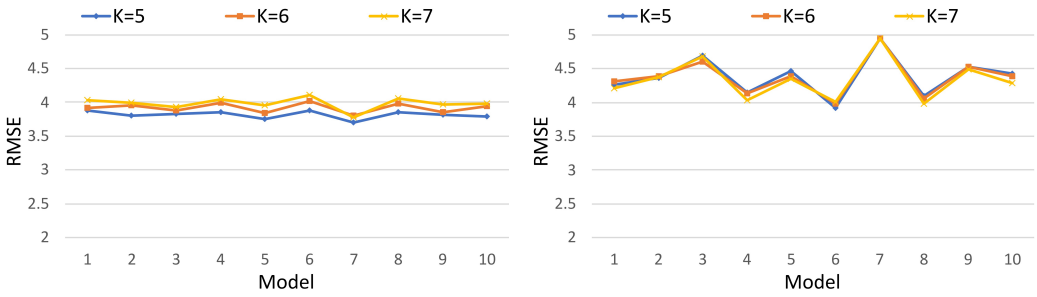


Figure 9. Comparison of KNN models, $K = 5, 6, 7$; RMSE for the training and test data of the 10 random models (75:25 ratio), with the Manhattan distance. (**a**) Training; (**b**) test.

Table 5. Statistical measures on the test data for KNN models; 10 models randomly generated with a 75:25 ratio by means of simple random sampling.

Model	St. Measures	Median	Mean	Confidence Interval (95%)
Manhattan distance $K = 5$	MAE	3.3871	3.3158	[3.1676, 3.4640]
	MAPE	17.7148	17.8784	[16.7483, 19.0085]
	RMSE	4.3912	4.3852	[4.1707, 4.5997]
Euclidean distance $K = 5$	MAE	3.4199	3.4384	[3.3320, 3.5448]
	MAPE	18.0707	18.5220	[17.6788, 19.3652]
	RMSE	4.4556	4.4921	[4.3254, 4.6588]
Manhattan distance $K = 6$	MAE	3.3981	3.3318	[3.1932, 3.4704]
	MAPE	17.5864	17.9849	[16.8763, 19.0935]
	RMSE	4.3845	4.3733	[4.1709, 4.5757]
Euclidean distance $K = 6$	MAE	3.4646	3.4592	[3.3369, 3.5815]
	MAPE	18.3423	18.6691	[17.6019, 19.7363]
	RMSE	4.4300	4.4808	[4.3001, 4.6615]
Manhattan distance $K = 7$	MAE	3.3550	3.3134	[3.1579, 3.4689]
	MAPE	17.4759	17.9372	[16.8324, 19.0420]
	RMSE	4.3177	4.3365	[4.1150, 4.5580]
Euclidean distance $K = 7$	MAE	3.4203	3.4474	[3.3160, 3.5788]
	MAPE	18.2648	18.6589	[17.5924, 19.7254]
	RMSE	4.4611	4.4920	[4.3151, 4.6689]

Table 6. Statistical measures for KNN models with the Manhattan distance.

Model	Measures	$K = 5$	$K = 7$
Without validation	R^2	0.8365	0.8115
	MAE	2.7562	2.9255
	MAPE	14.5033	15.5158
	RMSE	3.6246	3.8916
With validation (mean of 10 models)	R^2	0.8038	0.7930
	MAE	3.0006	3.1044
	MAPE	15.9617	16.6643
	RMSE	3.9694	4.0775
With validation (best model)	R^2	0.8120	0.7974
	MAE	2.9567	3.0597
	MAPE	15.6981	16.3948
	RMSE	3.8863	4.0340

3.2. MLP Neural Networks

The MLP is a commonly used ANN. In this section, we discuss its use to predict the compressive strength of LWAC. Although these neural networks can be designed using several hidden layers, the use of only one hidden layer improved the performance of these configurations. Hence, we focus here on the analysis of MLP neural networks with a single hidden layer. Several different activation functions were used: the identity function, the logistic sigmoid function, the hyperbolic tangent function and the ReLU function. The solvers used to minimise the squared error were SGD, L-BFGS and Adam. To simplify the models and prevent overfitting, Ridge or L2 regularisation was applied. Based on the L2 norm, this type of regularisation introduces an additional term to the loss function to penalise large weights and thus forces the weights of the network to take small values [51]. The values considered for the strength of the regularisation term were $\alpha = 0, 0.0001, 0.001, 0.01, 0.05, 0.08, 0.09, 0.1, 0.11, 0.12, 0.2, 0.5, 1, 2$. Note that $\alpha = 0$ means that no regularisation was applied. To determine the number of neurons for the hidden layer, architectures were designed with between one and 2^7 neurons in this layer. Figure 10 displays the change in the relative error as the number of neurons increases, where the L-BFGS optimiser and the ReLU activation function are used. This configuration gave the best performance for our problem. Table 7 summarises several goodness-of-fit measures for some of the models. A suitable value for α was found to be 0.1, as this value prevented overfitting and achieved more stable results, regardless of the number of neurons used (see Figure 10c). In this way, we avoid the problem in which a few neurons dominate the behaviour of the network, and we force non-informative features to have weights close to or equal to zero. We should point out that the ANN models considered here outperformed the best ANN architecture proposed in [6]. For example, with $\alpha = 0.1$, we obtained a mean RMSE value of 3.705 with 30 neurons in the hidden layer and a value of 3.6765 with 40 neurons (see Table 7). These values are lower than 3.745, which was the RMSE for the best model in [6]. Similar results were obtained with values of α very close to 0.1. Note that without Ridge regularisation, the models may be overfitted to the training data (see Figure 10a). This issue also occurs when very small values of α are applied ($\alpha \to 0$). In contrast, when $\alpha > 1$, the models are clearly underfitted (see Figure 10d). In fact, we obtained mean values of RMSE greater than four for $\alpha = 1$ and greater than 4.4 for $\alpha = 2$.

Table 7. Statistical measures for MLP networks; ReLU activation function, L-BFGS solver.

Number of Neurons, α	Measures	Mean of Test Sets	With Validation (Mean of 10 Models)
8, $\alpha = 0.01$	R^2	0.8014	0.8043
	MAE	3.0467	3.0404
	MAPE	16.7511	16.6353
	RMSE	3.9617	3.9628
30, $\alpha = 0.1$	R^2	0.8045	0.8290
	MAE	2.9960	2.8415
	MAPE	16.3343	15.4097
	RMSE	3.9328	3.7050
40, $\alpha = 0.1$	R^2	0.8032	0.8317
	MAE	2.9536	2.7956
	MAPE	15.9433	15.0795
	RMSE	3.9437	3.6765

3.3. Tree-Based Algorithms: Random Forest and Gradient-Boosted Tree Models

3.3.1. Random Forests

To construct the RF models, the dataset was also partitioned into training and test sets with a ratio of 75:25. This randomised process was repeated 10 times. Each of the random training datasets was then used to create an RF model, using bootstrap samples to build the trees. A specific percentage of samples was drawn from the training datasets to

train each base estimator. Once a random forest model had been obtained, its predictive accuracy and generalisability were analysed using the corresponding test dataset. Repeated *k*-fold cross-validation was also applied, and no appreciable differences between the two cross-validation procedures were found.

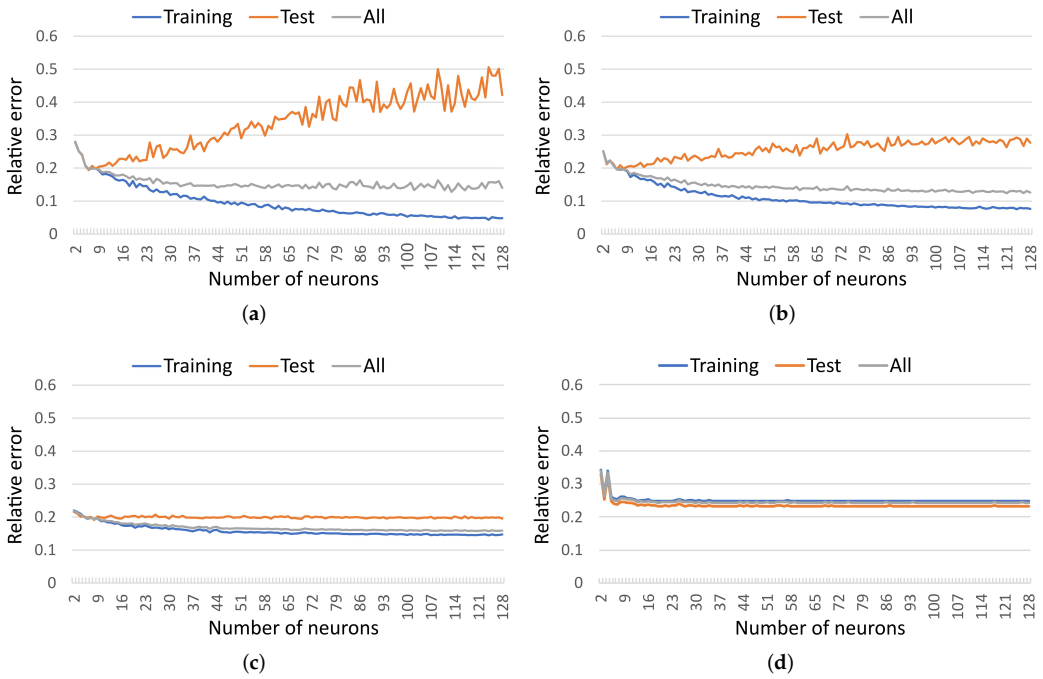


Figure 10. Relative error of the MLP networks for varying numbers of neurons in the hidden layer; number of iterations = 2000, ReLU activation function, L-BFGS solver. (**a**) $\alpha = 0$ (without L2 regularisation); (**b**) $\alpha = 0.01$; (**c**) $\alpha = 0.1$; (**d**) $\alpha = 2$.

To explore the influence of the number of samples used to train the base estimator, different percentages for the training set were considered. The results showed similar performance for percentages greater than or equal to 75% (see Figure 11). In addition, each RF model was created with varying numbers of trees, from two to 1000. No major differences were found when a number of trees greater than or equal to 10 was used (see Figure 12a). Moreover, as expected, no overfitting was detected as the number of trees increased. In the rest of the results discussed in this section, we use 100 trees to estimate each RF, as this number of trees gave optimal RF models (see Figure 12b). The splitting criteria used were the MSE and the MAE. Both gave the same performance.

To detect overfitting, we varied not only the maximum depth of the trees but also the minimum number of cases needed for parent and child nodes. Figure 13 illustrates the behaviour of the RMSE when the maximum depth of the trees was varied. In Figure 13a, the nodes could be expanded until all leaves are pure, while in Figure 13b, a minimum of 20 cases for parent nodes and 10 for child nodes are required to augment the branches. As can be seen from these figures, good results were obtained with a maximum depth of five. Figure 14 compares the RMSE for the training and test datasets of the 10 models with varying values of the minimum number of cases for parent and child nodes needed to increase the branches of the tree. Table 8 summarises several goodness-of-fit measures for the models shown in Figure 14. Acceptable results were obtained from pre-pruning the trees, where a minimum of 20 samples was required to split an internal node and 10 samples in the leaf or terminal nodes (model (c)). This model achieved a mean RMSE of 3.8434 and a mean MAE of 2.9809 for the test data. Although models (a) and (b) in Table 8 seem a priori to outperform model (c), decreasing the required cases for parent and child nodes could lead to overfitting; see Figure 14a with a minimum of two cases for internal nodes and one for terminal nodes and Figure 14b with a minimum of 10 cases for internal nodes and

five for terminal nodes. Increasing the cases required for parent and child nodes produced underfitting (see Figure 14d,e). Table 9 displays several goodness-of-fit measures for the models after the validation process. The parameters used to obtain these models were the same as those for model (c) in Table 8. In addition, various percentages of samples were used to train each base estimator, achieving a mean RMSE of between 3.6303 and 3.6784, a MAE of between 2.7808 and 2.8207, and a MAPE of between 15.5416% and 15.7916%.

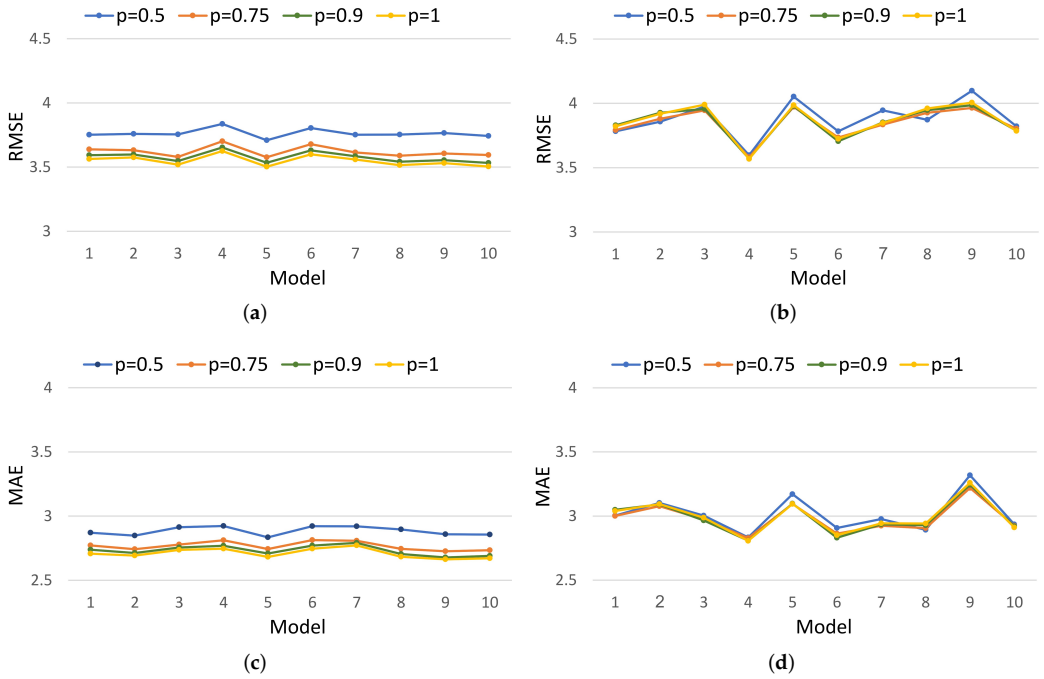


Figure 11. Behaviour of RF models for varying proportions of samples for training each base estimator; maximum depth = 5, minimum of 20 cases for parent nodes and 10 for child nodes, number of trees = 100. (**a**) RMSE, training dataset; (**b**) RMSE, test dataset; (**c**) MAE, training dataset; (**d**) MAE, test dataset.

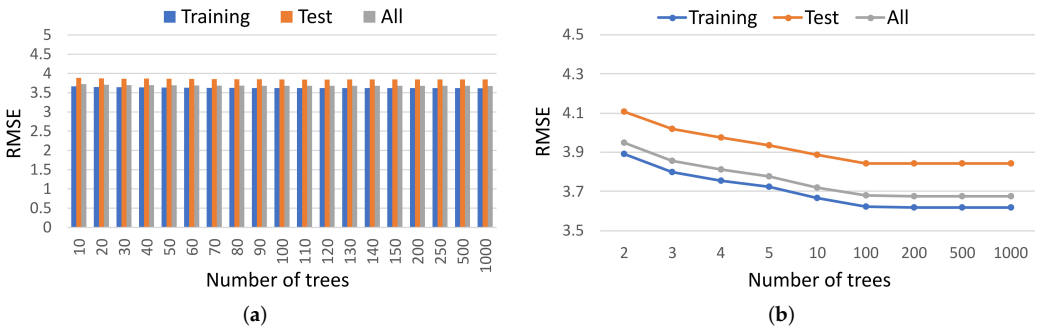


Figure 12. Behaviour of RF models for varying numbers of trees; maximum depth = 5, minimum of 20 cases for parent nodes and 10 for child nodes, proportion of samples for training each base estimator = 0.75. (**a**) RMSE, number of trees between 10 and 1000; (**b**) RMSE, number of trees between two and 1000.

As expected, RF outperformed the tree models described in [19] (see Table 3), which achieved a MAPE of approximately 16.22%, an RMSE of 3.808 and an MAE of 2.928 with the best model. In addition, the efficiency of RF was higher than that of both the KNN models (with a best mean RMSE of 3.9694) and the MLP neural networks (with a best mean RMSE of 3.6765); see Tables 6 and 7, respectively.

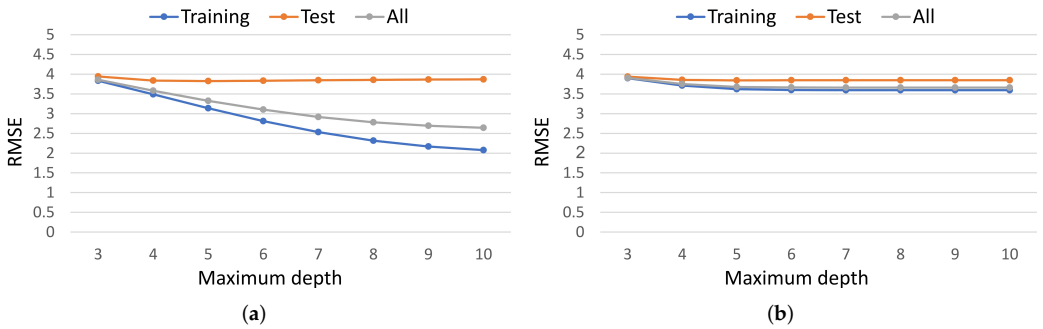


Figure 13. Behaviour of RF models for varying maximum depths; proportion of samples for training each base estimator = 0.75. (**a**) RMSE, minimum of two cases for parent nodes and one for child nodes; (**b**) RMSE, minimum of 20 cases for parent nodes and 10 for child nodes.

Table 8. Results for RF models; 10 models randomly generated with a 75:25 ratio by means of simple random sampling, maximum depth = 5, proportion of samples for training each base estimator = 0.75. Statistical measures for test data: (a) minimum of two cases for parent nodes and one for child nodes, (b) minimum of 10 cases for parent nodes and five for child nodes, (c) minimum of 20 cases for parent nodes and 10 for child nodes, (d) minimum of 40 cases for parent nodes and 20 for child nodes, (e) minimum of 50 cases for parent nodes and 25 for child nodes.

Model	St. Measures	Median	Mean	Confidence Interval (95%)
(a)	MAE	2.9328	2.9530	[2.8378, 3.0682]
	MAPE	16.2147	16.2340	[15.4197, 17.0483]
	RMSE	3.8316	3.8239	[3.6988, 3.9490]
(b)	MAE	2.9556	2.9557	[2.8570, 3.0544]
	MAPE	16.3162	16.4000	[15.5729, 17.2271]
	RMSE	3.8380	3.8279	[3.7241, 3.9317]
(c)	MAE	2.9492	2.9809	[2.8953, 3.0665]
	MAPE	16.7791	16.7259	[15.8576, 17.5942]
	RMSE	3.8571	3.8434	[3.7560, 3.9308]
(d)	MAE	3.1185	3.1390	[3.0210, 3.2570]
	MAPE	17.5578	17.7231	[16.7236, 18.7226]
	RMSE	4.0653	4.0443	[3.9077, 4.1809]
(e)	MAE	3.1682	3.1887	[3.0705, 3.3069]
	MAPE	17.7840	18.0240	[16.9883, 19.0597]
	RMSE	4.1343	4.1074	[3.9662, 4.2486]

3.3.2. Gradient-Boosted Tree-Based Models

The RF regressors studied in Section 3.3.1 are ensemble methods from the family of averaging methods. By contrast, in boosting methods, the base estimators are built sequentially, such that each decision tree attempts to reduce the error of the previous tree. In this section, we explore the use of two well-known boosting ensemble methods: GBR and extreme GBR (or XGBoost). To obtain gradient-boosted tree models using GBR, both the squared error and the absolute error were used as loss functions. In addition, the splitting criteria were the MSE and the MSE with Friedman's improvement score. When the absolute error loss function was used, the best splitting criterion was the Friedman MSE; however, the best results were obtained with the squared error loss function, which gave the same performance with both splitting criteria. Gradient-boosted tree models were obtained with varying numbers of boosting stages, from 10 to 1000. The maximum depth of the individual regression estimators was chosen as between one and five, and the best results were obtained with a value of three. For this maximum depth, Figure 15 illustrates the behaviour of the RMSE and MAE as the number of boosting iterations (added trees) increases. The learning rate was set to 0.02 in Figure 15a,b and to 0.05 in Figure 15c,d. The learning rate and the

number of estimators are two critical hyperparameters for this method [73]. These hyperparameters interact with each other in such a way that lower values of the learning rate require larger numbers of estimators to maintain similar RMSE and MAE values [55]. For a learning rate of 0.02, good results were obtained with 250 weak learners or trees (see Figure 15a,b). With this learning rate, a higher number of trees could lead to overfitting the model. A slight improvement was obtained with a learning rate of 0.05 and 130 weak learners. In this case, clear overfitting occurs with more than 200 added trees (see Figure 15c,d). In contrast, a small number of weak learners or a lower value of the learning rate could lead to underfitting of the model (see Figures 15 and 16). In addition, as the number of trees increases, the values of the learning rate that produce strong underfitting decrease. For example, with 250 weak learners, a learning rate of 0.005 or less caused underfitting of the models for our problem (see Figure 16c,d). With 130 weak learners, strong underfitting occurred for values less than or equal to 0.01 (see Figure 16a,b), while a learning rate greater than 0.1 produced overfitting in both cases.

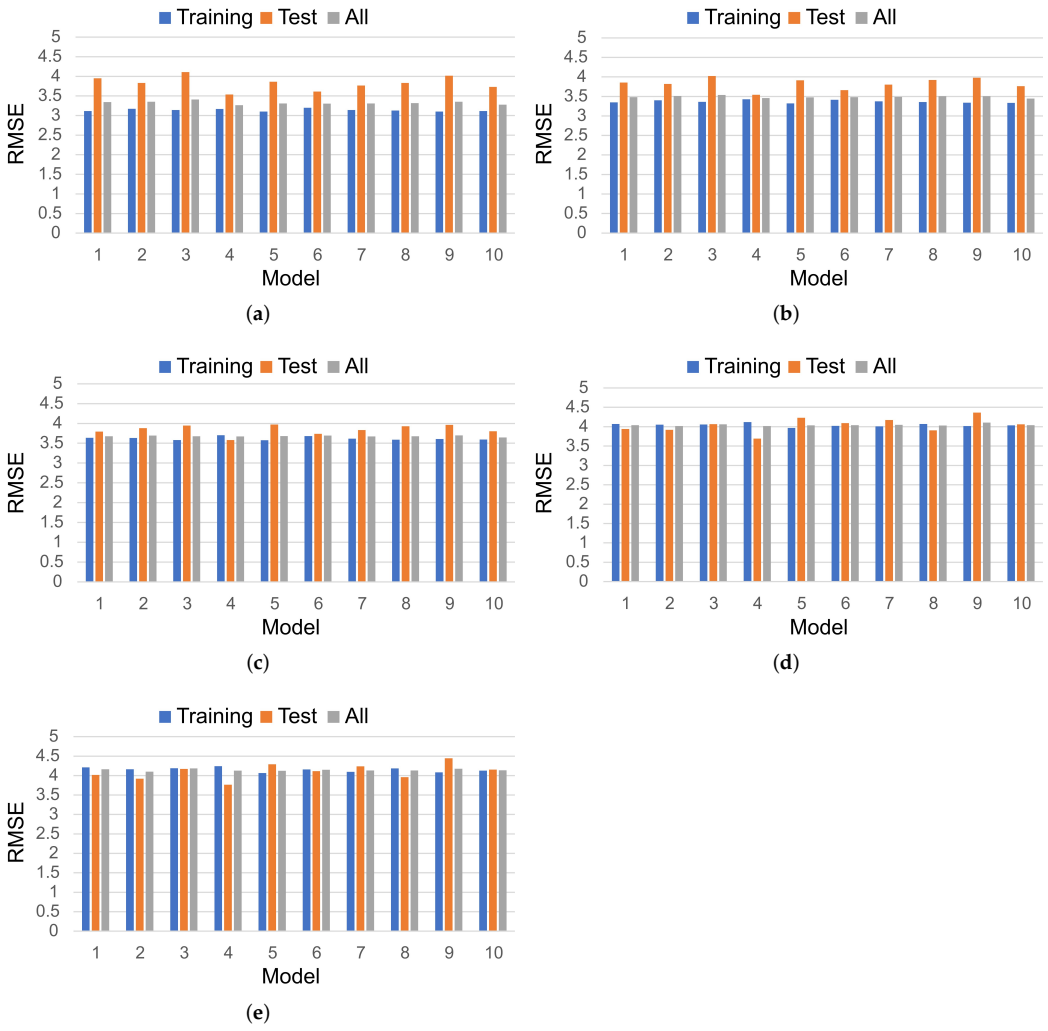


Figure 14. Behaviour of RF models for varying numbers of minimum cases for parent and child nodes; maximum depth = 5, percentage of samples for training each base estimator = 0.75. (**a**) Minimum of two cases for parent nodes and one for child nodes; (**b**) minimum of 10 cases for parent nodes and five for child nodes; (**c**) minimum of 20 cases for parent nodes and 10 for child nodes; (**d**) minimum of 40 cases for parent nodes and 20 for child nodes; (**e**) minimum of 50 cases for parent nodes and 25 for child nodes.

Table 9. Statistical measures for RF; minimum of 20 cases for parent nodes and 10 for child nodes, maximum depth = 5, p = proportion of samples for training each base estimator.

p	Measures	Mean of Test Sets	With Validation (Mean of 10 Models)
0.75	R^2	0.8135	0.8316
	MAE	2.9809	2.8207
	MAPE	16.7259	15.7916
	RMSE	3.8434	3.6784
0.9	R^2	0.8125	0.8343
	MAE	2.9867	2.7955
	MAPE	16.7081	15.6373
	RMSE	3.8534	3.6486
1	R^2	0.8119	0.8359
	MAE	2.9925	2.7808
	MAPE	16.7218	15.5416
	RMSE	3.8603	3.6303

From the above figures, we see that the best results were obtained with 130 boosting iterations (weak learners or trees) and a learning rate of between 0.04 and 0.06; a mean RMSE of 3.799, a mean MAE of 2.9338 and a mean MAPE of 16.2570% were achieved for the test datasets with a learning rate of 0.05.

In an attempt to reduce possible overfitting, the minimal cost-complexity pruning algorithm [61] was also used with the GBR algorithm. Figure 17 shows the influence of the value of the pruning parameter. In Figure 17a, a minimum of two cases for parent nodes and one for child nodes were considered, while Figure 17b shows a minimum of 20 cases for parent nodes and 10 for child nodes. In both cases, similar performance was obtained for a pruning parameter of 0.00001 or less, and this was comparable to the performance of the model when no pruning was carried out (i.e., a pruning parameter of zero). However, pruning parameters greater than 0.00001 caused underfitting of the models. Table 10 summarises several goodness-of-fit measures for the best configurations of hyperparameters found. Different proportions of samples were also used to train each base learner. The best gradient-boosted models achieved a mean RMSE of between 3.15 and 3.2979 and outperformed RF regardless of the proportion of samples extracted from the training dataset to train each base estimator (see Tables 9 and 10).

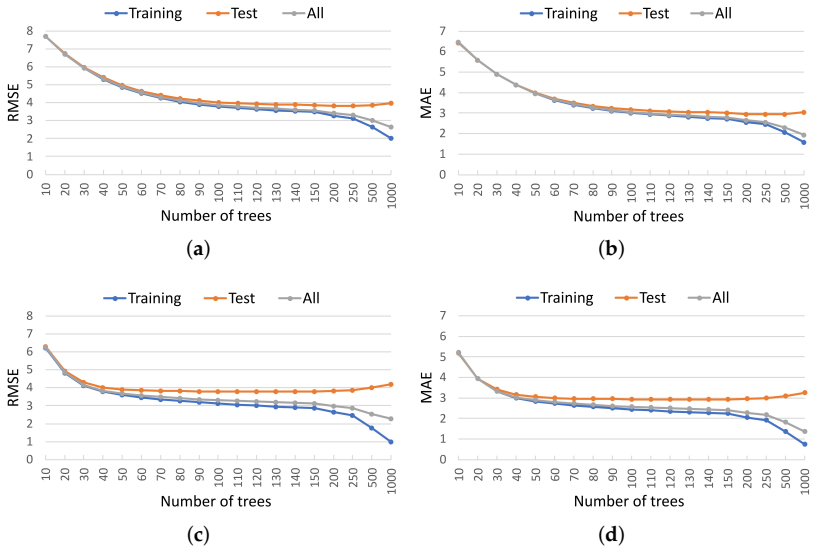


Figure 15. Gradient-boosted tree models with varying numbers of stages or built trees, for a minimum of two cases for parent nodes and one for child nodes; maximum depth = 3. (**a**) RMSE, learning rate = 0.02; (**b**) MAE, learning rate = 0.02; (**c**) RMSE, learning rate = 0.05; (**d**) MAE, learning rate = 0.05.

Table 10. Statistical measures for gradient-boosted tree models; learning rate = 0.05, number of estimators = 130, maximum depth = 3, p = proportion of samples for training each base learner; (a) minimum of two cases for parent nodes and one for child nodes, (b) minimum of 20 cases for parent nodes and 10 for child nodes.

Model	Measures	Mean of Test Sets	With Validation (Mean of 10 Models)
(a)	R^2	0.8185	0.8762
$p = 0.75$	MAE	2.9183	2.4472
	MAPE	16.2005	13.6766
	RMSE	3.7921	3.1528
(b)	R^2	0.8204	0.8646
$p = 0.75$	MAE	2.8988	2.5283
	MAPE	16.1394	14.1537
	RMSE	3.7725	3.2979
(a)	R^2	0.8182	0.8765
$p = 0.9$	MAE	2.9209	2.4396
	MAPE	16.2251	13.6751
	RMSE	3.7959	3.1500
(b)	R^2	0.8201	0.8590
$p = 0.9$	MAE	2.9032	2.5825
	MAPE	16.2201	14.4576
	RMSE	3.7747	3.3650
(a)	R^2	0.8178	0.8732
$p = 1.0$	MAE	2.9338	2.4690
	MAPE	16.2570	13.8063
	RMSE	3.7990	3.1907
(b)	R^2	0.8175	0.8632
$p = 1.0$	MAE	2.9318	2.5466
	MAPE	16.3024	14.2430
	RMSE	3.8021	3.3145

The gradient-boosted tree models discussed above were created using the Scikit-learn API. With the aim of optimising these models, we also used the XGBoost library, an optimised distributed gradient boosting library that allowed us to apply some regularisation techniques such as L1 and L2 regularisation [62] and dropout techniques [63]. The booster parameter sets the type of learner used in the algorithm: the gbtree booster allows for L1 and L2 regularisation, while the dart booster inherits the gbtree booster, but adds dropout techniques, allowing some trees to be dropped to solve the possible problem of overfitting. Both learners gave the same results when no dropout was added, and all instances of the training dataset were employed to train each base learner ($p = 1$). For $p < 1$, the dart learner slightly outperformed the gbtree learner, although the differences were not significant. Note that XGBoost does not support the hyperparameters related to the minimum number of samples required to split an internal or a leaf node; instead, the min_child_weight parameter can be controlled. This parameter represents the minimum sum of the instance weights needed for a child node. If the tree partitioning step results in a leaf node with a sum of the instance weights of less than min_child_weight, the building process will cease any further partitioning. In linear regression tasks, this simply corresponds to the minimum number of instances required for each node. This parameter was varied from 0 to 20, and the best performance was obtained for values of zero and one, which gave the same results. Figure 18 illustrates the behaviour of XGBoost with varying numbers of trees and a varying percentage of samples used to train each base learner. Neither L1 and L2 regularisation nor dropout techniques are included in these results. Under these conditions, XGBoost behaved analogously to the previous gradient-boosted tree models, in which the nodes could be expanded until all the leaves were pure. That is, both algorithms gave a similar performance as the number of trees increased. More specifically,

for a learning rate of 0.05, both models gave good results for numbers of base estimators of between 90 and 130. However, XGBoost produced more underfitting for small numbers of trees. When setting the best configurations of hyperparameters, we found that XGBoost slightly outperformed the previous gradient-boosted tree model. For example, with 130 weak learners, a learning rate of 0.05 and 75% of samples used to train each base estimator, XGBoost achieved a mean RMSE of 3.1402, a mean MAE of 2.4375 and a mean MAPE of 13.6244%, i.e., better results than those shown in Table 10 (see model (a) in Tables 10 and 11). To analyse the prediction accuracy of XGBoost depending on the type of regularisation applied, the parameters used to control the L1 and L2 regularisation (α and λ, respectively) were varied in the set $\{0, 0.0001, 0.001, 0.01, 0.1, 0.2\}$ and the dropout rate was varied in the set $\{0, 0.01, 0.05, 0.1\}$. The best results were obtained for $\alpha = 0$, $\lambda = 0.001$, and a dropout rate of zero (see Figures 19 and 20). The prediction accuracy of these models was comparable to the results obtained without these types of regularisation. More specifically, under the above conditions, XGBoost achieved a mean RMSE of 3.1396, a mean MAE of 2.4371 and a mean MAPE of 13.6238%. For the test datasets, the mean values were 3.7812, 2.9020 and 16.0921%, respectively (see Table 11). Note that both the gradient-boosted tree models and the extreme gradient-boosted tree models outperformed RF.

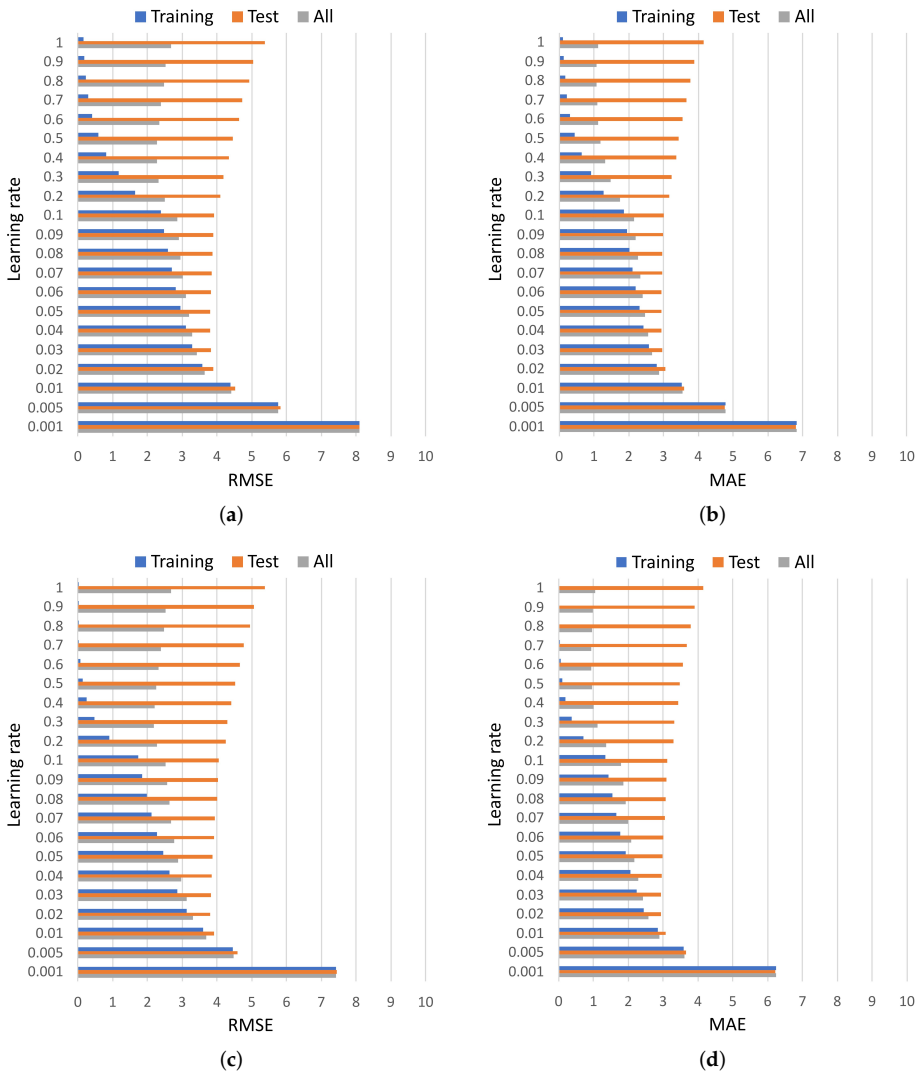
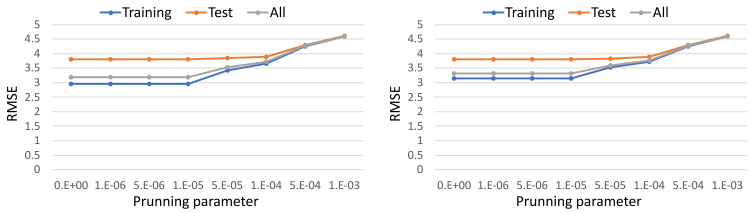
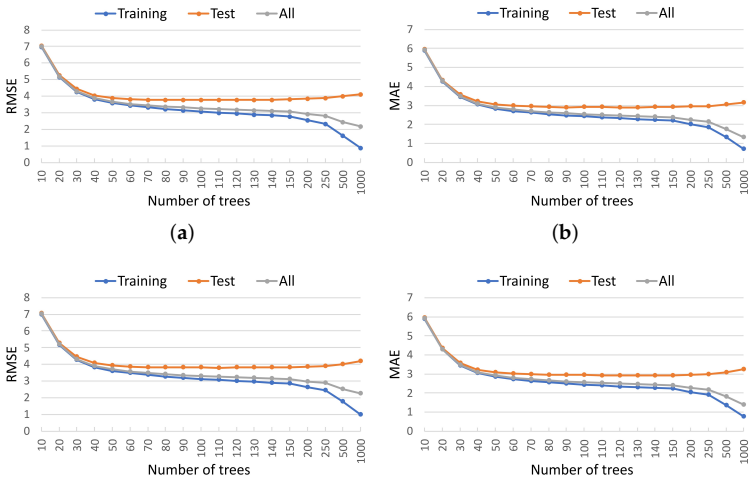


Figure 16. Gradient-boosted tree models with varying learning rates, for a minimum of two cases for parent nodes and one for child nodes; maximum depth = 3. (**a**) RMSE, number of estimators = 130; (**b**) MAE, number of estimators = 130; (**c**) RMSE, number of estimators = 250; (**d**) MAE, number of estimators = 250.



(**a**) (**b**)

Figure 17. Gradient-boosted tree models with varying pruning parameters; learning rate = 0.05, proportion of samples for training each base learner = 1. (**a**) RMSE. Minimum of two cases for parent nodes and one for child nodes; (**b**) RMSE. Minimum of 20 cases for parent nodes and 10 for child nodes.



(**a**) (**b**)

(**c**) (**d**)

Figure 18. Results for XGBoost models; 10 models randomly generated with a 75:25 ratio by means of simple random sampling, maximum depth = 3, learning rate = 0.05, minimum sum of instance weight needed in a child node = 1, p = proportion of samples for training each base estimator. (**a**) RMSE, $p = 0.75$; (**b**) MAE, $p = 0.75$; (**c**) RMSE, $p = 1$; (**d**) MAE, $p = 1$.

Table 11. Results for XGBoost models; learning rate = 0.05, number of estimators = 130, maximum depth = 3, p = proportion of samples for training each base learner, minimum sum of instance weight needed in a child node = 1.

Model	Measures	Mean of Test Set	With Validation (Mean of 10 Models)
$p = 0.75$	R^2	0.8195	0.8772
$\alpha = 0$	MAE	2.9042	2.4375
$\lambda = 0$	MAPE	16.1032	13.6244
	RMSE	3.7821	3.1402
$p = 0.75$	R^2	0.8196	0.8773
$\alpha = 0$	MAE	2.9020	2.4371
$\lambda = 0.001$	MAPE	16.0921	13.6238
	RMSE	3.7812	3.1396
$p = 0.9$	R^2	0.8172	0.8761
$\alpha = 0$	MAE	2.9261	2.4408
$\lambda = 0$	MAPE	16.2213	13.6526
	RMSE	3.806	3.1545
$p = 0.9$	R^2	0.8175	0.8761
$\alpha = 0$	MAE	2.9256	2.4416
$\lambda = 0.001$	MAPE	16.2244	13.6555
	RMSE	3.8027	3.1549
$p = 1.0$	R^2	0.8172	0.8731
$\alpha = 0$	MAE	2.9356	2.4693
$\lambda = 0$	MAPE	16.2774	13.8164
	RMSE	3.8051	3.1926
$p = 1.0$	R^2	0.8167	0.8730
$\alpha = 0$	MAE	2.9393	2.4698
$\lambda = 0.001$	MAPE	16.3016	13.8291
	RMSE	3.8104	3.1936

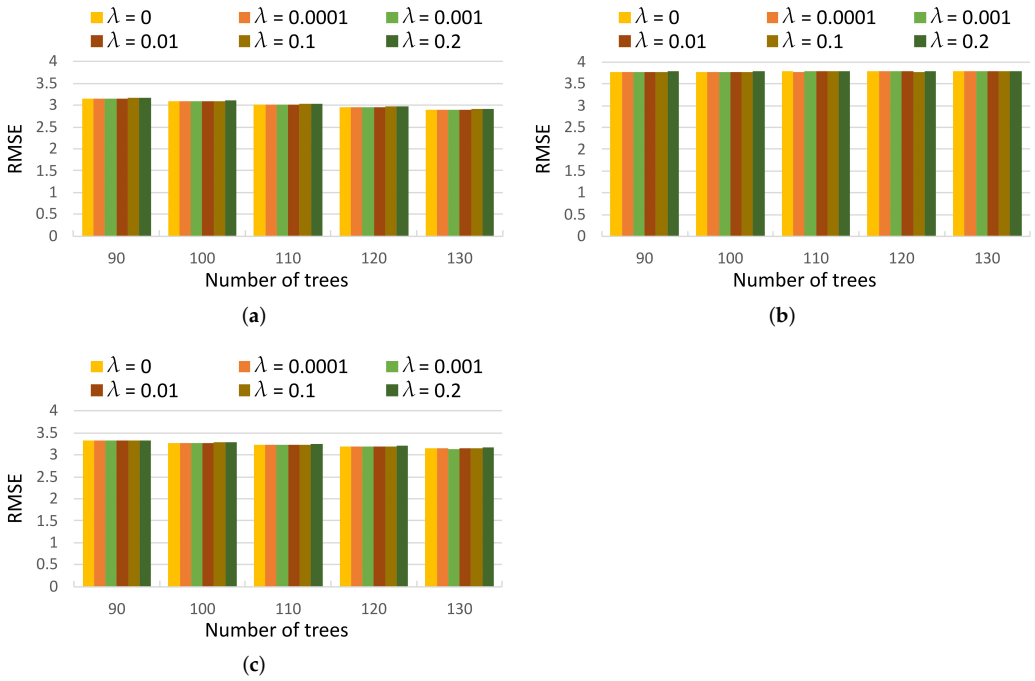


Figure 19. Results for XGBoost models with varying λ; learning rate = 0.05, number of estimators = 130, maximum depth = 3, $p = 0.75$, minimum sum of instance weight needed in a child node = 1, dropout rate = 0, $\alpha = 0$. (**a**) RMSE, training dataset; (**b**) RMSE, test dataset; (**c**) RMSE, all datasets.

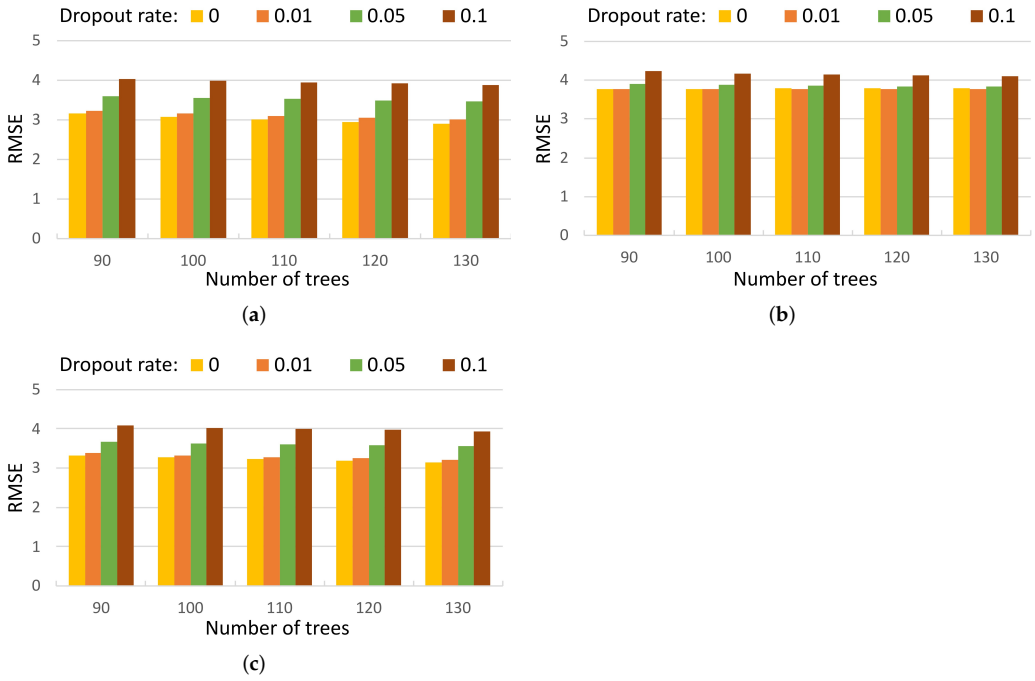


Figure 20. Results for XGBoost models with varying dropout rate; learning rate = 0.05, number of estimators = 130, maximum depth = 3, $p = 0.75$, minimum sum of instance weight needed in a child node = 1, $\alpha = 0$, $\lambda = 0.001$. (**a**) RMSE, training dataset; (**b**) RMSE, test dataset; (**c**) RMSE, all datasets.

3.4. Support Vector Regression

SVR methods were also studied in terms of predicting the compressive strength of LWAC. For this purpose, we used linear, polynomial, RBF and sigmoid kernels, and different values of C, ϵ and γ were explored. The regularisation parameter C was varied between 0.1 and 100, and ϵ was drawn from the set $\{0.00001, 0.0001, 0.001, 0.002, 0.003, 0.004, 0.005,$

$0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4\}$. The values for γ were $\gamma = \frac{1}{d}$ and $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$, where d is the number of input variables and $\mathrm{Var}(X)$ is the variance of the input dataset. The use of linear, polynomial and sigmoid kernels failed to improve the performance of the RBF kernel, giving models with RMSE values greater than four. We, therefore, focus here on an analysis of the SVR models with RBF kernel. Figure 21 illustrates the behaviour of the SVR models with RBF kernel for various values of ϵ for the epsilon tube. The best results were obtained with $\epsilon \leq 0.05$ for both choices of γ. For $\epsilon > 0.1$, the models were underfitted.

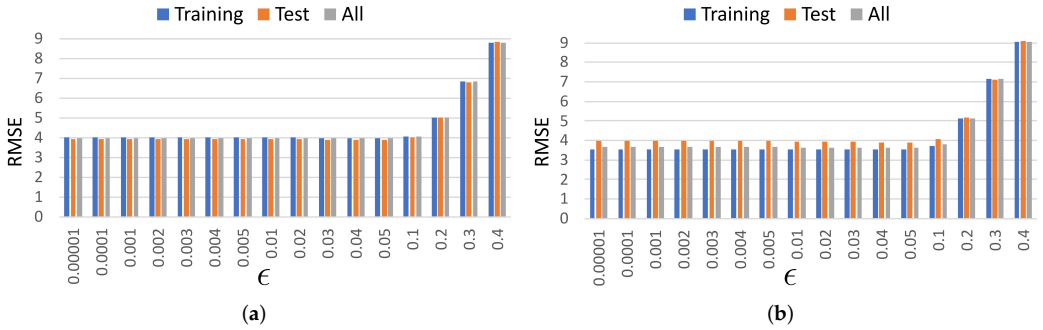


Figure 21. RMSE for the SVR models; RBF kernel, $C = 1$. (**a**) $\gamma = \frac{1}{d}$; (**b**) $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$.

The influence of the parameter C is illustrated in Figure 22 for good values of ϵ. Figure 22a,b shows the behaviour of the RMSE for $\gamma = \frac{1}{d}$, and Figure 22c,d shows the results for $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$. Different behaviour was observed depending on the value of γ. For $\gamma = \frac{1}{d}$, the best results were obtained with a value for C of between two and nine; however, for $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$, a lower value of C was needed to avoid overfitting. In this case, the optimal values of C were between 0.3 and 0.7, and this model outperformed the one based on $\gamma = \frac{1}{d}$. Table 12 displays several metrics, with C in this range, for this value of γ.

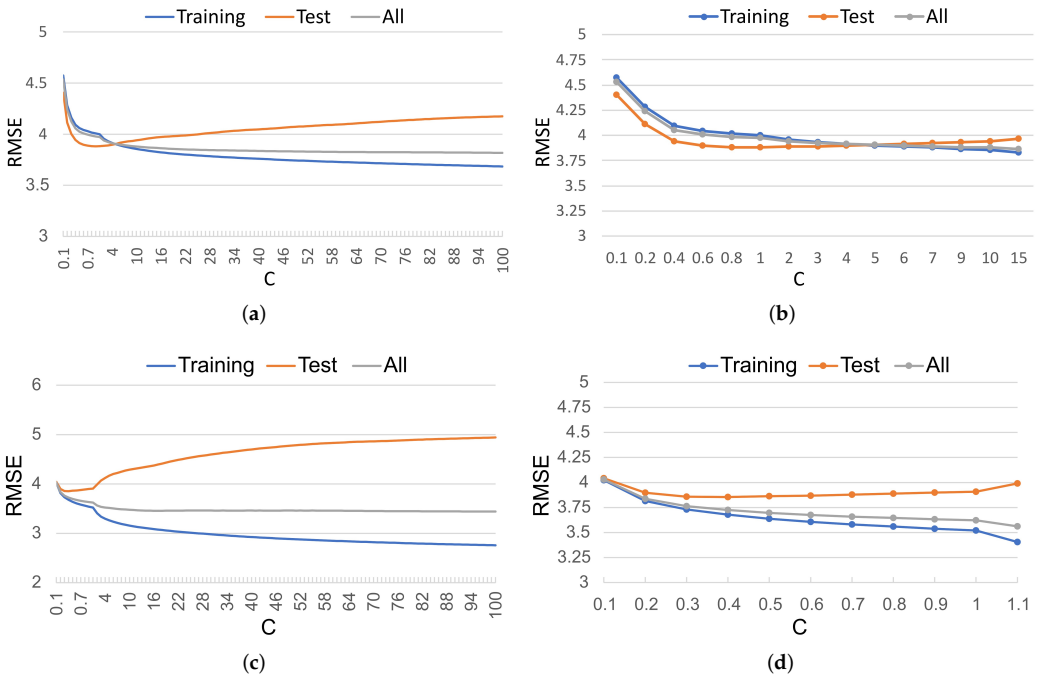


Figure 22. RMSE for the SVR models for varying values of C; RBF kernel. (**a**) $\gamma = \frac{1}{d}$, $\epsilon = 0.05$, $0.1 \leq C \leq 100$; (**b**) $\gamma = \frac{1}{d}$, $\epsilon = 0.05$, $0.1 \leq C \leq 15$; (**c**) $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$, $\epsilon = 0.04$, $0.1 \leq C \leq 100$; (**d**) $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$, $\epsilon = 0.04$, $0.1 \leq C \leq 1.1$.

Table 12. Statistical measures on the test data for SVR models; 10 models randomly generated with a 75:25 ratio by means of simple random sampling, RBF kernel, $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$, $\epsilon = 0.04$.

C	Measures	Mean of Test Sets	With Validation (Mean of 10 Models)
0.3	R^2	0.8120	0.8237
	MAE	3.0000	2.8778
	MAPE	16.2627	15.5709
	RMSE	3.8586	3.7637
0.4	R^2	0.8124	0.8272
	MAE	2.9944	2.852
	MAPE	16.1798	15.4092
	RMSE	3.8543	3.7252
0.5	R^2	0.8115	0.8299
	MAE	3.0004	2.832
	MAPE	16.182	15.2909
	RMSE	3.8639	3.6970
0.6	R^2	0.8111	0.8134
	MAE	2.9986	2.8152
	MAPE	16.1617	15.1992
	RMSE	3.8679	3.6751
0.7	R^2	0.8101	0.8333
	MAE	3.0035	2.8016
	MAPE	16.1789	15.1239
	RMSE	3.8783	3.6591

The performance of these models was similar to that of the RF models, with a mean RMSE of between 3.8543 and 3.8783 for the test sets and between 3.6591 and 3.7637 for the overall dataset.

3.5. Weighted Average Ensemble Models

We developed a weighted average ensemble (WAE) method to predict the compressive strength of LWAC using the tree-based ensemble methods and the SVR method. The RF, SVR and XGBoost models were combined to give a more powerful model than each of these approaches alone. The triplet $(\alpha_1, \alpha_2, \alpha_3)$ denotes the weights associated with each of these models, respectively. These parameters were varied from zero to one, such that $\alpha_1 + \alpha_2 + \alpha_3 = 1$. The predictions from the WAE method were obtained following the flowchart shown in Figure 23 as a weighted sum of the results from the RF, SVR and XGBoost models, using the weights α_1, α_2 and α_3. For the cross-validation procedure, the dataset was split into training and test sets with a ratio of 75:25, using the same seeds as in the previous experiments. This randomised process was repeated 10 times. For each of these sets, the training process was carried out to tune the hyperparameters associated with these methods. The initialisation of the training procedure was conducted using good values for each individual model. To search for the optimal weights, the regularisation was controlled by means of the C parameter of the SVR method. In addition, the behaviour of this ensemble method was explored for varying proportions of the samples (p) used to train each base learner. The results showed good performance for $p = 0.75$ (see Figure 24).

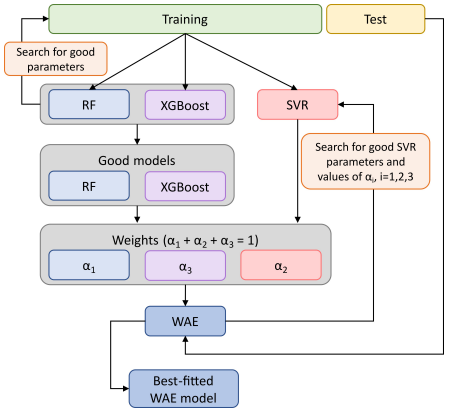


Figure 23. Flowchart of the WAE method.

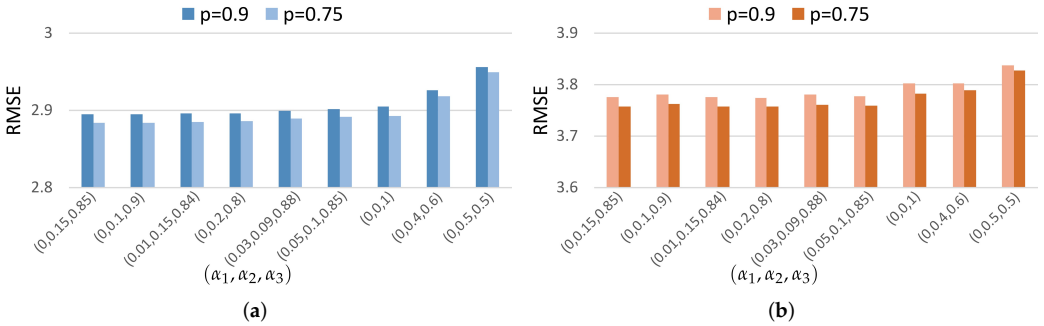


Figure 24. Results for WAE models; $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$, p = proportion of samples for training each base learner, learning rate = 0.05, $\epsilon = 0.04$, $C = 6$. (**a**) RMSE, training dataset; (**b**) RMSE, test dataset.

The WAE models were also rated based on the values of their goodness-of-fit measures, using both the discrete and continuous rankings described in Section 2.1.6 (see Table 13). Based on these rankings, Table 14 summarises several goodness-of-fit measures for the best configurations of hyperparameters found. For a tolerance error of 0.04, a good compromise between the accuracy of the models on the training sets and the accuracy on the test sets was obtained for $0.6 \leq C \leq 6$. For these configurations, suitable models were obtained with $0 \leq \alpha_1 \leq 0.15$, $0.65 \leq \alpha_3 \leq 1$, and $\alpha_2 = 1 - \alpha_1 - \alpha_3$, with a mean RMSE of between 3.75 and 3.757, a mean MAE of between 2.872 and 2.877, and a mean MAPE of approximately 15.8% for the test datasets. These values were lower than those obtained from each individual estimator. We conclude that a good choice of weights for this ensemble method allows it to outperform the XGBoost models.

For setting competitive models, we first identified the three WAE models with the lowest values of the overall continuous ranking; these were models (6), (4) and (5), with an overall rank of approximately 7.89 (see Table 13). We then analysed the corresponding training and test ratings to detect possible overfitting. Similar values were obtained for the training and test datasets, indicating that no overfitting was detected. Next, we analysed the discrete ranking. In this case, higher overall ranks were obtained for models (6), (2) and (4). Since model (2) obtained the worst discrete rank in the training process, we concluded that the settings of models (4) and (6) are preferable. As can be seen from Table 14, both models performed well on the test dataset, with similar values of goodness-of-fit. Model (4) had a mean determination coefficient of 0.8201, a mean RMSE of 3.757, a mean MAE of 2.8768 and a mean MAPE of 15.845%, for the test dataset, while for model (6), the values were 0.822, 3.7562, 2.8755 and 15.8056%, respectively.

Table 13. Ranking analysis of WAE models; D: discrete rank, C: continuous rank, learning rate = 0.05, number of estimators = 130, maximum depth = 3, $p = 0.75$, $\epsilon = 0.04$, $\lambda = 0.001$, $C = 6$.

WAE Model	Dataset	$R^2/1-R^2$ D/C	MAE D/C	MAPE D/C	RMSE D/C	Total Rank D/C	Overall Rank D/C
(1) $\alpha_1 = 0$, $\alpha_2 = 0.4$, $\alpha_3 = 0.60$	Training	2/0.9925	3/0.9882	6/0.9779	2/0.9962	13/3.9548	21/7.9505
	Test	1/1	1/1	5/0.9957	1/1	8/3.9957	
(2) $\alpha_1 = 0.15$, $\alpha_2 = 0.2$, $\alpha_3 = 0.65$	Training	3/0.9878	2/0.9923	2/0.9904	3/0.9940	10/3.9645	32/7.9222
	Test	5/0.9796	6/0.9928	6/0.9953	5/0.9900	22/3.9577	
(3) $\alpha_1 = 0.25$, $\alpha_2 = 0.15$, $\alpha_3 = 0.60$	Training	1/1	1/1	1/1	1/1	4/4	25/7.9594
	Test	6/0.9790	5/0.9934	4/0.9972	6/0.9898	21/3.9594	
(4) $\alpha_1 = 0$, $\alpha_2 = 0.15$, $\alpha_3 = 0.85$	Training	6/0.9691	5/0.9847	4/0.9858	6/0.9844	21/3.9240	29/7.8930
	Test	3/0.9829	2/0.9945	1/1	2/0.9916	8/3.9690	
(5) $\alpha_1 = 0.01$, $\alpha_2 = 0.15$, $\alpha_3 = 0.84$	Training	4/0.9700	4/0.9852	3/0.9863	5/0.9849	16/3.9264	26/7.8949
	Test	2/0.9829	3/0.9944	2/0.9998	3/0.9914	10/3.9685	
(6) $\alpha_1 = 0$, $\alpha_2 = 0.2$, $\alpha_3 = 0.80$	Training	5/0.9700	6/0.9835	5/0.9823	4/0.9850	20/3.9208	35/7.8861
	Test	4/0.9823	4/0.9941	3/0.9975	4/0.9914	15/3.9653	

Table 14. Results for WAE models; learning rate = 0.05, number of estimators = 130, maximum depth = 3, $p = 0.75$, $\epsilon = 0.04$, $\lambda = 0.001$, $C = 6$.

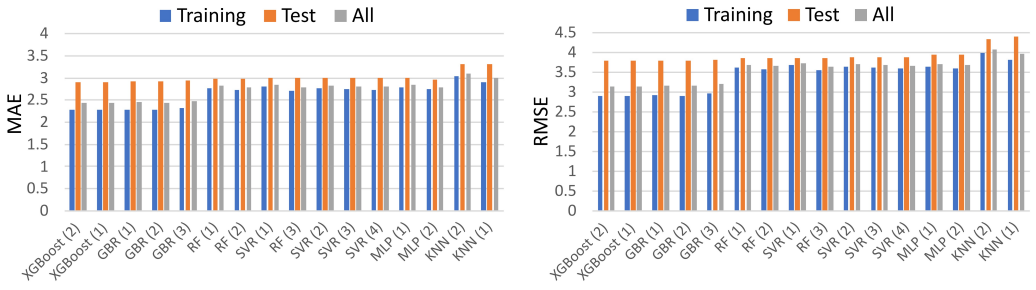
Model	Measures	Mean of Test Sets	With Validation (Mean of 10 Models)
(2)	R^2	0.8225	0.8769
$\alpha_1 = 0.15$	MAE	2.8720	2.4278
$\alpha_2 = 0.20$	MAPE	15.7700	13.4316
$\alpha_3 = 0.65$	RMSE	3.7509	3.1438
(4)	R^2	0.8201	0.8783
$\alpha_1 = 0$	MAE	2.8768	2.4159
$\alpha_2 = 0.15$	MAPE	15.8450	13.4064
$\alpha_3 = 0.85$	RMSE	3.7570	3.1261
(5)	R^2	0.8219	0.8783
$\alpha_1 = 0.01$	MAE	2.8764	2.4167
$\alpha_2 = 0.15$	MAPE	15.8422	13.4105
$\alpha_3 = 0.84$	RMSE	3.7564	3.1270
(6)	R^2	0.822	0.8782
$\alpha_1 = 0$	MAE	2.8755	2.4135
$\alpha_2 = 0.20$	MAPE	15.8056	13.3628
$\alpha_2 = 0.80$	RMSE	3.7562	3.1272

3.6. Comparison of Models

Figures 25–27 present a comparison of the above models. As can be seen from these figures, the MLP, RF and SVR models achieved similar results, with best mean RMSEs of 3.6765, 3.6303 and 3.6591, respectively, and best mean MAPEs of 15.0795%, 15.5416% and 15.1239% (see Table 15). In addition, their efficiencies were higher than those of the KNN models (with a best mean RMSE of 3.9694, a best mean MAPE of 15.9617%, and a best MAE of 3.0006). Although these three models gave better performance than the ANN models studied in [6] and the MLR and RT models explored in [19], none of them outperformed the models obtained in this study using the GBR, XGBoost, and WAE methods (see Figure 27).

In addition, the comparison of the models treated herein with the models obtained in previous research works [6,19] also showed that the MLR [19] and KNN models were the worst in terms of predicting the compressive strength of LWAC. KNN showed slightly lower performance than the best tree model in [19] (RMSE = 3.808) and the best ANN model in [6] (RMSE = 3.745). The KNN models based on the Manhattan distance only succeeded in outperforming the MLR models (with a best mean RMSE of 4.332), while the KNN models based on the Euclidean distance, with a best mean RMSE of 4.4808, were worse than the MLR models.

A comparison between GBR and XGBoost models showed that the XGBoost models achieved the smallest mean values for the RMSE and MAE on both the training and test datasets (see Figures 25 and 26). With suitable hyperparameter settings, the XGBoost models outperformed GBR models and achieved a mean determination coefficient of approximately 0.8773, a mean RMSE of 3.1396, a mean MAE of 2.4371, and a mean MAPE of 13.6238% (see Table 15 and Figure 27).



(**a**) (**b**)

Figure 25. Comparison of the KNN, MLP, RF, GBR, XGBoost and SVR models (see Table 15 for the *x*-axis labels). (**a**) MAE; (**b**) RMSE.

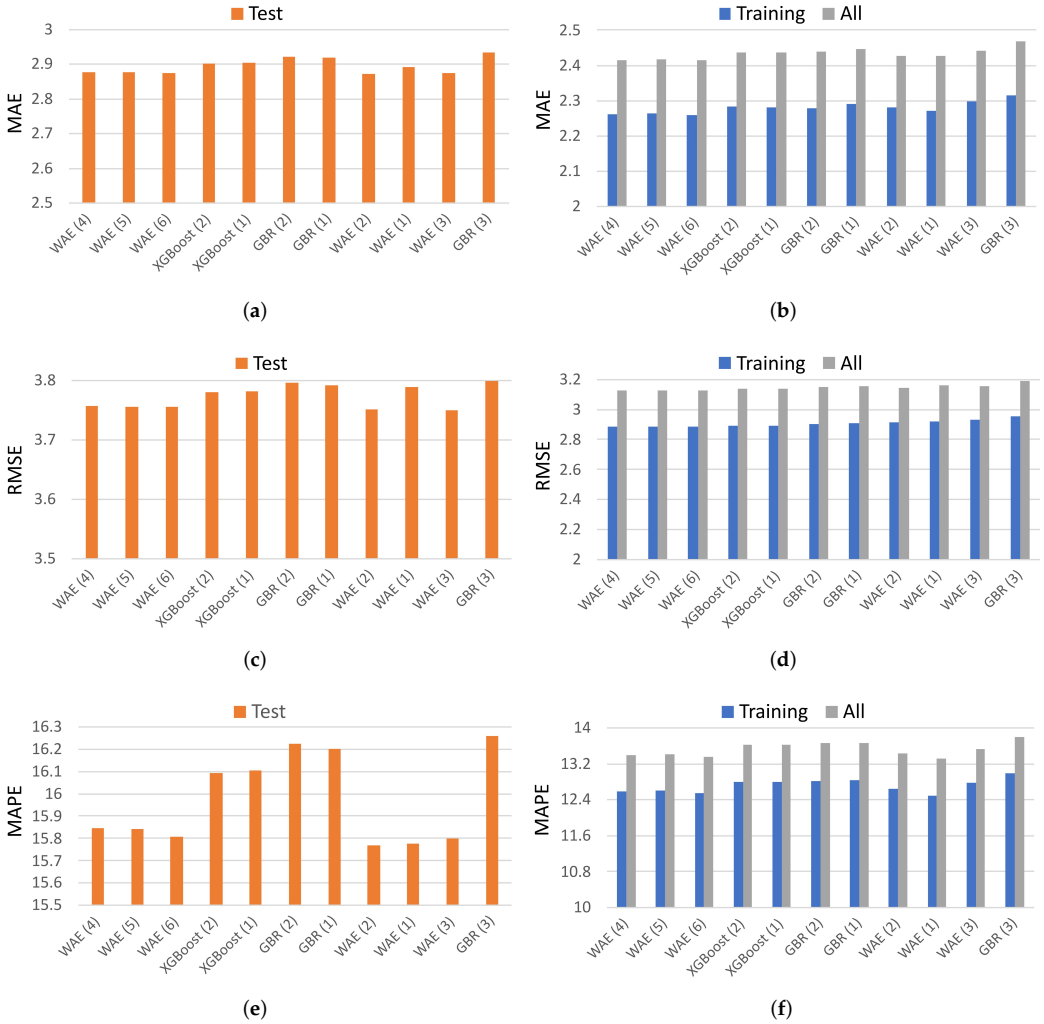


Figure 26. Comparison of the GBR, XGBoost and WAE models (see Table 15 for the x-axis labels). (**a**) MAE, test dataset; (**b**) MAE, training and all dataset; (**c**) RMSE, test dataset; (**d**) RMSE, training and all dataset; (**e**) MAPE, test dataset; (**f**) MAPE, training and all dataset.

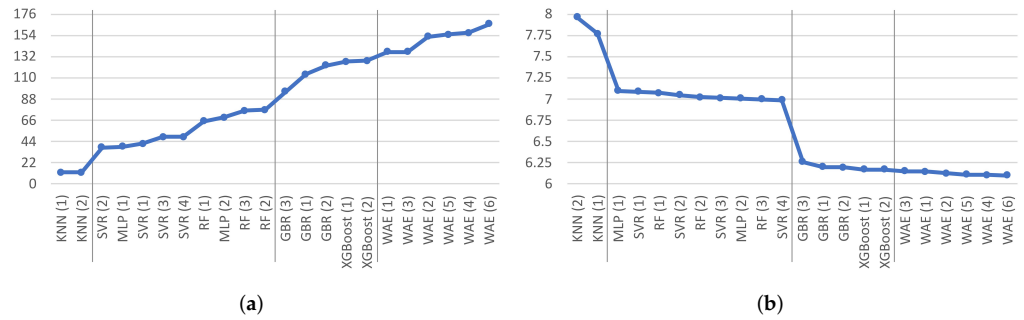


Figure 27. Ranking of the models of Table 15. (**a**) Discrete ranking; (**b**) continuous ranking.

The mean values of the performance measures for the best machine learning techniques (GBR, XGBoost and WAE) are shown in Table 16. This table and the analysis of the ranking of the best GBR, XGBoost and WAE models in Table 17 indicate that with appropriate values of the weight parameters, the proposed WAE models based on a combination of the RF, SVR, and XGBoost outperformed the GBR and XGBoost models. Note that the overall rank of the WAE model is higher than those of the GBR and XGBoost models. This is also true for the total rank for both the training and test datasets. For the results in Table 16, we

used $\alpha_2 = 0.2$ and $\alpha_3 = 0.8$. This model gave a mean determination coefficient of 0.8782, a mean RMSE of 3.1272, a mean MAE of 2.4135 and a mean MAPE of 13.3628%. Similar results were obtained for a combination of the SVR and XGBoost models with α_2 between 0.15 and 0.2, and α_3 between 0.8 and 0.85.

Table 15. Comparison of models based on statistical measures (mean of the 10 models); md = maximum depth, nt = number of trees, p = proportion of samples for training each base estimator, dr = dropout ratio, $^{(*)}$ best model.

Method	Parameters	R^2	MAE	MAPE	RMSE
MLR $^{(*)}$ [19]	-	0.766	3.396	18.86	4.332
ANN $^{(*)}$ [6]	6 neurons	0.825	2.897	15.85	3.745
RT $^{(*)}$ [19]	CHAID	0.820	2.928	16.22	3.808
KNN (1)	Manhattan distance, $K = 5$	0.8038	3.0006	15.9617	3.9694
KNN (2)	Manhattan distance, $K = 7$	0.7930	3.1044	16.6643	4.0775
MLP (1)	30 neurons, L-BFGS, ReLU, $\alpha = 0.1$	0.8290	2.8415	15.4097	3.7050
MLP (2)	40 neurons, L-BFGS, ReLU, $\alpha = 0.1$	0.8317	2.7956	15.0795	3.6765
RF (1)	$md = 5$, $nt = 100$, $p = 0.75$	0.8316	2.8207	15.7916	3.6784
RF (2)	$md = 5$, $nt = 100$, $p = 0.9$	0.8343	2.7955	15.6373	3.6486
RF (3)	$md = 5$, $nt = 100$, $p = 1$	0.8359	2.7808	15.5416	3.6303
GBR (1)	$md = 3$, $nt = 130$, $p = 0.75$	0.8762	2.4472	13.6766	3.1528
GBR (2)	$md = 3$, $nt = 130$, $p = 0.9$	0.8765	2.4396	13.6751	3.1500
GBR (3)	$md = 3$, $nt = 130$, $p = 1$	0.8732	2.4690	13.8063	3.1907
XGBoost (1)	$md = 3$, $nt = 130$, $\alpha = 0$, $\lambda = 0$, $dr = 0$, $p = 0.75$	0.8772	2.4375	13.6244	3.1402
XGBoost (2)	$md = 3$, $nt = 130$, $\alpha = 0$, $\lambda = 0.001$, $dr = 0$, $p = 0.75$	0.8773	2.4371	13.6238	3.1396
SVR (1)	$C = 0.4$, $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$, $\epsilon = 0.04$	0.8272	2.8520	15.4092	3.7252
SVR (2)	$C = 0.5$, $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$, $\epsilon = 0.04$	0.8299	2.8320	15.2909	3.6970
SVR (3)	$C = 0.6$, $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$, $\epsilon = 0.04$	0.8134	2.8152	15.1992	3.6751
SVR (4)	$C = 0.7$, $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$, $\epsilon = 0.04$	0.8333	2.8016	15.1239	3.6591
WAE (1)	$C = 6$, $p = 0.75$, $\alpha_1 = 0$, $\alpha_2 = 0.4$, $\alpha_3 = 0.60$	0.8757	2.4259	13.3133	3.1598
WAE (2)	$C = 6$, $p = 0.75$, $\alpha_1 = 0.15$, $\alpha_2 = 0.2$, $\alpha_3 = 0.65$	0.8769	2.4278	13.4316	3.1438
WAE (3)	$C = 6$, $p = 0.75$, $\alpha_1 = 0.25$, $\alpha_2 = 0.15$, $\alpha_3 = 0.60$	0.8760	2.4415	13.5316	3.1557
WAE (4)	$C = 6$, $p = 0.75$, $\alpha_1 = 0$, $\alpha_2 = 0.15$, $\alpha_3 = 0.85$	0.8783	2.4159	13.4064	3.1261
WAE (5)	$C = 6$, $p = 0.75$, $\alpha_1 = 0.01$, $\alpha_2 = 0.15$, $\alpha_3 = 0.84$	0.8783	2.4167	13.4105	3.1270
WAE (6)	$C = 6$, $p = 0.75$, $\alpha_1 = 0$, $\alpha_2 = 0.2$, $\alpha_3 = 0.80$	0.8782	2.4135	13.3628	3.1272

An analysis of the importance of each variable on the final predicted response for the different models is shown in Figures 28 and 29. More specifically, Figure 28 compares the importance of each variable in the RF, GBR and XGBoost models using the normalised feature importance based on the mean decrease in impurity. For the RF and GBR models, the most informative variables are the LWA particle density, the LWAC fixed density and the experimental dry density. However, the order of importance of these variables differs between the two models. The number of important features is higher for the XGBoost models, and these are the experimental dry density, the segregation index, the *P*-wave velocity, the concrete laying time and the LWA particle density. Figure 29 shows the importance of each variable in the MLP, KNN and SVR models based on feature permutation. The permutation feature importance is the decrease in the score for a model when a single feature value is randomly shuffled. As can be seen from this figure, the MLP and SVR models have the same three most important variables as the RF and GBR models, while the KNN model has only two of these three variables as the most important: the LWA particle density and the LWAC fixed density. This is consistent with the fact that the KNN models are worse than the MLP, SVR and tree-based models. In addition, the XGBoost models (and hence the WAE models) have more informative variables and outperform all other techniques discussed in this work. Regarding the computational cost, KNN and SVR required approximately 0.04 s for training and testing each model, while RF and GBR required between 0.08 and 0.12 s, and XGBoost required approximately 0.4 s. The WAE algorithm had the highest computational cost and complexity and required a mean time of 1.1 s; however, it obtained the best performance.

Table 16. Comparison of GBR, XGBoost and WAE models; 10 models randomly generated with a 75:25 ratio by means of simple random sampling (see Table 15 for the description of the models).

Method	Dataset	St. Measures	Median	Mean	Confidence Interval (95%)
GBR (2)	Training	R^2	0.8952	0.8954	[0.8934, 0.8974]
		MAE	2.2738	2.2792	[2.2533, 2.3051]
		MAPE	12.7757	12.8251	[12.6434, 13.0068]
		RMSE	2.9005	2.9009	[2.8671, 2.9347]
	Test	R^2	0.8144	0.8182	[0.8078, 0.8286]
		MAE	2.9413	2.9209	[2.7964, 3.0454]
		MAPE	15.9138	16.2251	[15.3890, 17.0612]
		RMSE	3.8291	3.7959	[3.6467, 3.9441]
	All	R^2	0.8765	0.8765	[0.8741, 0.8789]
		MAE	2.4403	2.4396	[2.4197, 2.4595]
		MAPE	13.6819	13.6751	[13.5575, 13.7927]
		RMSE	3.1504	3.1500	[3.1185, 3.1815]
XGBoost (2)	Training	R^2	0.8954	0.8960	[0.8942, 0.8978]
		MAE	2.2936	2.2822	[2.2524, 2.3120]
		MAPE	12.7911	12.8010	[12.6456,12.9564]
		RMSE	2.8853	2.8926	[2.8605,2.9247]
	Test	R^2	0.8160	0.8196	[0.8111, 0.8281]
		MAE	2.8887	2.9020	[2.7886, 3.0154]
		MAPE	15.9980	16.0921	[15.3427, 16.8415]
		RMSE	3.8046	3.7812	[3.6523, 3.9101]
	All	R^2	0.8773	0.8773	[0.8754, 0.8792]
		MAE	2.4384	2.4371	[2.4181, 2.4561]
		MAPE	13.6637	13.6238	[13.4818, 13.7658]
		RMSE	3.1394	3.1396	[3.1148, 3.1644]
WAE (6)	Training	R^2	0.8967	0.8965	[0.8950, 0.8980]
		MAE	2.2643	2.2595	[2.2340, 2.2850]
		MAPE	12.5287	12.5486	[12.3974, 12.6998]
		RMSE	2.8769	2.8857	[2.8593, 2.9121]
	Test	R^2	0.8185	0.8220	[0.8137, 0.8303]
		MAE	2.8447	2.8755	[2.7697, 2.9813]
		MAPE	15.7647	15.8056	[15.0947, 16.5165]
		RMSE	3.7892	3.7562	[3.6275, 3.8849]
	All	R^2	0.8785	0.8782	[0.8761, 0.8803]
		MAE	2.4112	2.4135	[2.3987, 2.4283]
		MAPE	13.3920	13.3628	[13.2487, 13.4769]
		RMSE	3.1240	3.1272	[3.0997, 3.1547]

Table 17. Discrete ranking analysis of GBR, XGBoost and WAE models (see Table 15 for the description of the models).

Model	Dataset	R^2	MAE	MAPE	RMSE	Total Rank	Overall Rank
GBR (2)	Training	1	2	1	1	5	9
	Test	1	1	1	1	4	
XGBoost (2)	Training	2	1	2	2	7	15
	Test	2	2	2	2	8	
WAE (6)	Training	3	3	3	3	12	24
	Test	3	3	3	3	12	

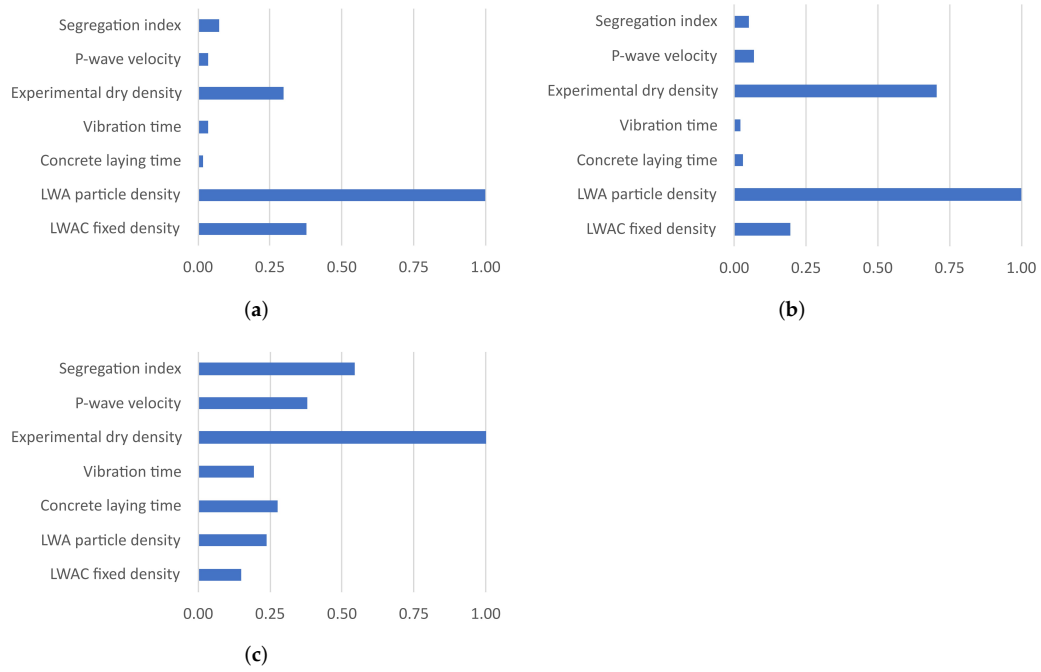


Figure 28. Normalised feature importance for RF, GBR and XGBoost models. (**a**) RF model; (**b**) GBR model; (**c**) XGBoost model.

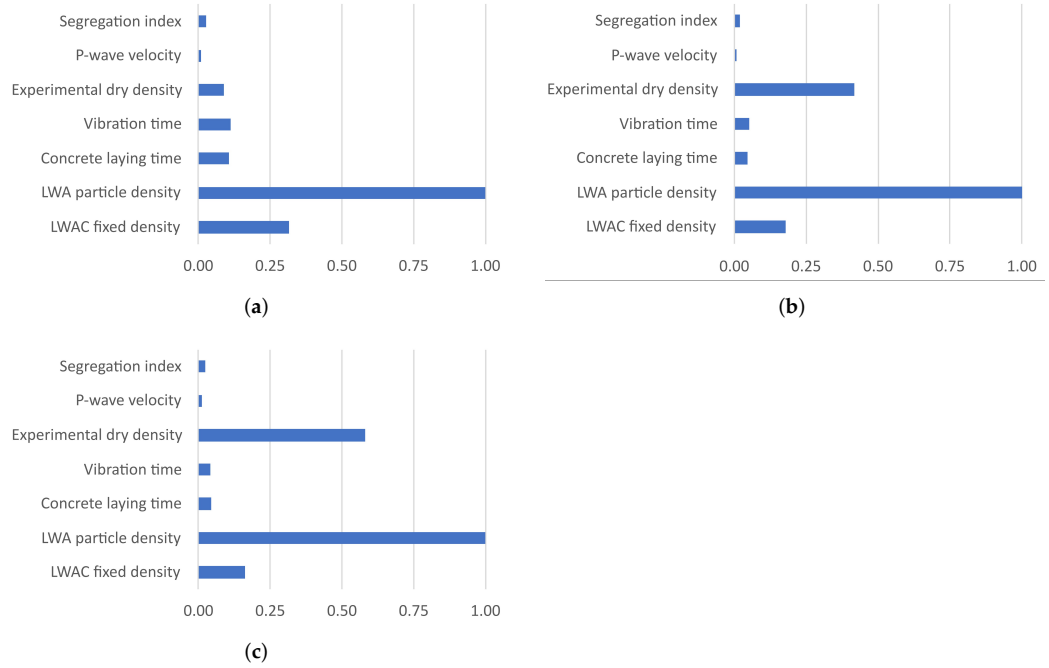


Figure 29. Normalised feature permutation importance for KNN, SVR and MLP models. (**a**) KNN model; (**b**) SVR model; (**c**) MLP model.

4. Conclusions

In this work, several well-known machine learning techniques have been explored with the aim of selecting the best in terms of predicting the compressive strength of segregated LWAC with expanded clay. The attributes involved in this problem were the particle density of the LWA, the laying time of the concrete, the vibration time, the experimental dry density of the specimen obtained after 28 days, the *P*-wave velocity, and the segregation index based on the *P*-wave velocities proposed in [6]. The KNN, RF, GBR, XGBoost and

SVR algorithms were implemented, and their performance was explored for varying values of the hyperparameters involved. In addition, a weighted ensemble method was devised that combined the RF, SVR and XGBoost models by using weight parameters α_1, α_2, α_3, respectively. In an attempt to optimise the benefits of the variance-reducing effects of this ensemble learning technique, the amount of regularisation was explored through the parameter C of the SVR technique. The efficiency of these models was compared, both with each other and with models proposed in previous works [6,19]. Two cross-validation procedures were applied, involving Monte Carlo and repeated k-fold cross-validation, and no significant differences in the results were found.

The behaviour of these methods indicated that the KNN models based on the Manhattan distance outperformed the MLR models presented in [19]. However, this technique gave the poorest performance of those implemented in this work. All of the other techniques studied in this work (MLP, RF, SVR, GBR, XGBoost and WAE) outperformed the machine learning techniques used in previous works (MLR, ANN and RT) [6,19]. The MLP, RF and SVR models behaved similarly, with best mean RMSEs of 3.6765, 3.6303 and 3.6591, respectively, and mean MAPEs of 15.0795%, 15.5406%, and 15.1239%. However, the RF models were the most stable (least influenced by the data splitting process), and a ranking analysis indicated that their best model was above the MLP models. The same performance was obtained for the MSE and MAE splitting criteria, and the best results were achieved with an ensemble of 100 trees, a maximum depth of five, and a proportion of samples for training each base estimator greater than or equal to 0.75. A good compromise between accuracy and the prevention of overfitting was found based on pre-pruning the tree with a minimum of 20 samples for child nodes and 10 for parent nodes. Suitable settings for the hyperparameters when training the MLP networks were the use of the L-BFGS optimiser, a ReLU activation function, and a strength of 0.1 for Ridge regularisation, which gave the best results with one hidden layer and a number of neurons between 30 and 40. However, with regard to the stability of these models, we observed that the MLP method was the most strongly influenced by the choice of the training and test data split. In relation to SVR, good configurations of parameters were obtained with the RBF kernel, $\epsilon = 0.04$, a regularisation parameter C of between 0.3 and 0.7, and $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$. These models exhibited similar stability to those obtained with the best GBR, XGBoost and WAE techniques.

The optimised version of the GBR technique, XGBoost, which allows for some regularisation techniques to be applied, outperformed the other GBR models. Suitable hyperparameter settings were the use of the dart booster, a strength of 0.001 for L2 regularisation, a maximum depth of three, and between 90 and 130 boosting stages. These XGBoost models achieved a mean determination coefficient of approximately 0.8773, a mean RMSE of 3.1396, a mean MAE of 2.4371, and a mean MAPE of 13.6238%, whereas the values obtained with the GBR model were 0.8765, 3.15, 2.4396 and 13.6751%, respectively.

However, our proposed WAE method achieved the best prediction results for the compressive strength of LWAC, with weights of $0 \leq \alpha_1 \leq 0.15$, $0.65 \leq \alpha_3 \leq 1$ and $\alpha_2 = 1 - \alpha_1 - \alpha_3$. We postulate that WAE methods combining SVR and XGBoost, with a value of α_2 between 0.15 and 0.20, and a value of α_3 between 0.80 and 0.85 represent the best strategies. With these weight parameters, our WAE models achieved a mean determination coefficient of about 0.878, a mean RMSE of about 3.127, a mean MAE of about 2.415 and a mean MAPE of 13.4%. This is coherent with the results of our ranking analysis of the best models for each technique, which identified the WAE models with weights in these intervals as the best options, followed by the XGBoost and GBR models, and a third group consisting of the RF, SVR and MLP models. This ranking finished with the ANN, RT, KNN and MLR models, in this order.

Our analysis of the importance of the input variables in each machine learning technique showed that for the RF, MLP, SVR and GBR models, the most informative variables were the LWA particle density, the LWAC fixed density and the experimental dry density, although the order of importance of some of these variables differed. However, for the worst model (KNN), only two of these three variables were the most important (the LWA

particle density and the LWAC fixed density). The XGBoost models, and hence the WAE models, had more informative features and outperformed all other techniques discussed in this work. These two methods included more important variables than the other techniques in terms of predicting the compressive strength of segregated LWAC (experimental dry density, segregation index, *P*-wave velocity, concrete lying time and LWA particle density). The most important of these five variables were the experimental dry density, the segregation index and the *P*-wave velocity.

The use of non-destructive techniques in the evaluation of the quality of concrete allows for monitoring of the condition of concrete and ensuring its proper production during construction. The large variability in the aggregates, types of LWA, cement, target density of concrete, and the additions and additives used for its production makes it difficult to correlate non-destructive techniques with strength quality. The results of our study demonstrate that computational intelligence models are reliable for use in predicting the compressive strength of LWAC using ultrasonic pulse velocity and would allow for continuous verification of the quality of the LWAC, both for industrialised construction and during the placement of concrete, since they include more variables in the learning process.

Author Contributions: Conceptualisation, V.M., J.P. and A.J.T.-A.; methodology, V.M. and J.P.; software, V.M., H.P., and J.P.; validation, V.M., H.P., J.P. and A.J.T.-A.; formal analysis, V.M. and H.P.; investigation, V.M., H.P. and J.P.; data curation, V.M. and A.J.T.-A.; writing—original draft preparation, V.M., H.P. and J.P.; writing—review and editing, V.M., J.P. and A.J.T.-A.; supervision, V.M. and J.P.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by MCIN/AEI/10.13039/501100011033, grant PID2021-123627OB-C55 and by "ERDF A way of making Europe".

Data Availability Statement: The data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LWAC	Lightweight aggregate concrete
LWA	Lightweight aggregate
ANN	Artificial neural network
MLP	Multilayer perceptron
RMSE	Root mean square error
MLR	Multiple linear regression
RT	Regression tree
CHAID	Chi-squared automatic interaction detector
CRT	Classification and regression trees
NLR	Nonlinear regression
FRP	Fiber-reinforced plastic
SVR	Support vector regression
GBR	Gradient boosting regressor
XGBoost	Extreme gradient boosting regressor
KNN	*K*-nearest neighbours
RF	Random forest
GPR	Gaussian progress regression
MAE	Mean absolute error
WAE	Weighted average ensemble
ReLU	Rectified linear unit
MSE	Mean squared error
SGD	Stochastic gradient descent
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno
RBF	Radial basis function
MAPE	Mean absolute percentage error

References

1. Chandra, S.; Berntsson, L. *Lightweight Aggregate Concrete*; Elsevier: Amsterdam, The Netherlands, 2002.
2. Haller, T.; Beuntner, N.; Gutsch, H.; Thienel, K.C. Challenges on pumping infra-lightweight concrete based on highly porous aggregates. *J. Build. Eng.* **2023**, *65*, 105761. [CrossRef]
3. Agrawal, Y.; Gupta, T.; Sharma, R.; Panwar, N.L.; Siddique, S. A Comprehensive Review on the Performance of Structural Lightweight Aggregate Concrete for Sustainable Construction. *Constr. Mater.* **2021**, *1*, 39–62. . constrmater1010003. [CrossRef]
4. Wu, T.; Yang, X.; Wei, H.; Liu, X. Mechanical properties and microstructure of lightweight aggregate concrete with and without fibers. *Constr. Build. Mater.* **2019**, *199*, 526–539. [CrossRef]
5. Xu, Z.; Li, Z. Numerical method for predicting flow and segregation behaviors of fresh concrete. *Cem. Concr. Compos.* **2021**, *123*, 104150. [CrossRef]
6. Tenza-Abril, A.J.; Villacampa, Y.; Solak, A.M.; Baeza-Brotons, F. Prediction and sensitivity analysis of compressive strength in segregated lightweight concrete based on artificial neural network using ultrasonic pulse velocity. *Constr. Build. Mater.* **2018**, *189*, 1173–1183. [CrossRef]
7. Tenza-Abril, A.; Benavente, D.; Pla, C.; Baeza-Brotons, F.; Valdes-Abellan, J.; Solak, A. Statistical and experimental study for determining the influence of the segregation phenomenon on physical and mechanical properties of lightweight concrete. *Constr. Build. Mater.* **2020**, *238*, 117642. [CrossRef]
8. Leemann, A.; Münch, B.; Gasser, P.; Holzer, L. Influence of compaction on the interfacial transition zone and the permeability of concrete. *Cem. Concr. Res.* **2006**, *36*, 1425–1433. [CrossRef]
9. Solak, A.M.; Tenza-Abril, A.J.; García-Vera, V.E. Adopting an image analysis method to study the influence of segregation on the compressive strength of lightweight aggregate concretes. *Constr. Build. Mater.* **2022**, *323*, 126594. [CrossRef]
10. Bogas, J.A.; Gomes, M.G.; Real, S.; Pontes, J. Ultrasonic pulse velocity used to predict the compressive strength of structural sand lightweight concrete. In Proceedings of the First International Conference on Construction Materials and Structures, Singapore, 4–6 November 1987; IOS Press: Amsterdam, The Netherlands, 2014; pp. 293–304. [CrossRef]
11. Navarrete, I.; Lopez, M. Estimating the segregation of concrete based on mixture design and vibratory energy. *Constr. Build. Mater.* **2016**, *122*, 384–390. [CrossRef]
12. Ke, Y. Characterization of the Mechanical Behavior of Lightweight Aggregate Concretes: Experiment and Modelling. Ph.D Thesis, Université de Cergy-Pontoise, Cergy-Pontoise, France, 2008.
13. Ke, Y.; Beaucour, A.; Ortola, S.; Dumontet, H.; Cabrillac, R. Influence of volume fraction and characteristics of lightweight aggregates on the mechanical properties of concrete. *Constr. Build. Mater.* **2009**, *23*, 2821–2828. . 2009.02.038. [CrossRef]
14. Solak, A.M.; Tenza-Abril, A.J.; Baeza-Brotons, F.; Benavente, D. Proposing a New Method Based on Image Analysis to Estimate the Segregation Index of Lightweight Aggregate Concretes. *Materials* **2019**, *12*, 3642. [CrossRef]
15. Hemmatian, A.; Jalali, M.; Naderpour, H.; Nehdi, M.L. Machine learning prediction of fiber pull-out and bond-slip in fiber-reinforced cementitious composites. *J. Build. Eng.* **2023**, *63*, 105474. [CrossRef]
16. Chen, Z.; Zhang, L.; Li, K.; Xue, X.; Zhang, X.; Kim, B.; Li, C.Y. Machine-learning prediction of aerodynamic damping for buildings and structures undergoing flow-induced vibrations. *J. Build. Eng.* **2023**, *63*, 105374. [CrossRef]
17. Sajan, K.C.; Bhusal, A.; Gautam, D.; Rupakhety, R. Earthquake damage and rehabilitation intervention prediction using machine learning. *Eng. Fail. Anal.* **2023**, *144*, 106949. [CrossRef]
18. Li, Z.; Yoon, J.; Zhang, R.; Rajabipour, F.; Srubar III, W.V.; Dabo, I.; Radlińska, A. Machine learning in concrete science: Applications, challenges, and best practices. *Npj Comput. Mater.* **2022**, *8*, 127. [CrossRef]
19. Migallón, V.; Navarro-González, F.J.; Penadés, J.; Villacampa, Y. Parallel approach of a Galerkin-based methodology for predicting the compressive strength of the lightweight aggregate concrete. *Constr. Build. Mater.* **2019**, *219*, 56–68. [CrossRef]
20. Kass, G.V. An Exploratory Technique for Investigating Large Quantities of Categorical Data. *J. R. Stat. Soc. C Appl. Stat.* **1980**, *29*, 119–127. [CrossRef]
21. Biggs, D.; Ville, B.D.; Suen, E. A method of choosing multiway partitions for classification and decision trees. *J. Appl. Stat.* **1991**, *18*, 49–62. [CrossRef]
22. Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. *Classification and Regression Trees*, 1st ed.; Routledge: London, UK, 1984. [CrossRef]
23. Kewalramani, M.A.; Gupta, R. Concrete compressive strength prediction using ultrasonic pulse velocity through artificial neural networks. *Automat. Constr.* **2006**, *15*, 374–379. [CrossRef]
24. Tavakkol, S.; Alapour, F.; Kazemian, A.; Hasaninejad, A.; Ghanbari, A.; Ramezanianpour, A.A. Prediction of lightweight concrete strength by categorized regression, MLR and ANN. *Comput. Concr.* **2013**, *12*, 151–167. [CrossRef]
25. Charhate, S.; Subhedar, M.; Adsul, N. Prediction of Concrete Properties Using Multiple Linear Regression and Artificial Neural Network. *J. Soft Comput. Civ. Eng.* **2018**, *2*, 27–38. [CrossRef]
26. Kalman Šipoš, T.; Miličević, I.; Siddique, R. Model for mix design of brick aggregate concrete based on neural network modelling. *Constr. Build. Mater.* **2017**, *148*, 757–769. [CrossRef]
27. Deshpande, N.; Londhe, S.; Kulkarni, S. Modeling compressive strength of recycled aggregate concrete by Artificial Neural Network, Model Tree and Non-linear Regression. *Int. J. Sustain. Built Environ.* **2014**, *3*, 187–198. [CrossRef]
28. Behnood, A.; Olek, J.; Glinicki, M.A. Predicting modulus elasticity of recycled aggregate concrete using M5′ model tree algorithm. *Constr. Build. Mater.* **2015**, *94*, 137–147. [CrossRef]

29. Karbassi, A.; Mohebi, B.; Rezaee, S.; Lestuzzi, P. Damage prediction for regular reinforced concrete buildings using the decision tree algorithm. *Comput. Struct.* **2014**, *130*, 46–56. [CrossRef]
30. Huang, L.; Chen, J.; Tan, X. BP-ANN based bond strength prediction for FRP reinforced concrete at high temperature. *Eng. Struct.* **2022**, *257*, 114026. [CrossRef]
31. Wang, X.L.; Zha, X.X.; Zhang, X.C. Bonding properties of FRP bars and concrete at high temperature. *J. Harbin. Inst. Technol.* **2013**, *45*, 8–15.
32. El-Gamal, S. Bond strength of glass fiber-reinforced polymer bars in concrete after exposure to elevated temperatures. *J. Reinf. Plast. Compos.* **2014**, *33*, 2151–2163. [CrossRef]
33. Özkal, F.M.; Polat, M.; Yağan, M.; Öztürk, M.O. Mechanical properties and bond strength degradation of GFRP and steel rebars at elevated temperatures. *Constr. Build. Mater.* **2018**, *184*, 45–57. [CrossRef]
34. Erdal, H.I. Two-level and hybrid ensembles of decision trees for high performance concrete compressive strength prediction. *Eng. Appl. Artif. Intell.* **2013**, *26*, 1689–1697. [CrossRef]
35. Nguyen, H.; Vu, T.; Vo, T.P.; Thai, H.T. Efficient machine learning models for prediction of concrete strengths. *Constr. Build. Mater.* **2021**, *266*, 120950. [CrossRef]
36. Chou, J.S.; Pham, A.D. Enhanced artificial intelligence for ensemble approach to predicting high performance concrete compressive strength. *Constr. Build. Mater.* **2013**, *49*, 554–563. [CrossRef]
37. Mousavi, S.M.; Aminian, P.; Gandomi, A.H.; Alavi, A.H.; Bolandi, H. A new predictive model for compressive strength of HPC using gene expression programming. *Adv. Eng. Softw.* **2012**, *45*, 105–114. [CrossRef]
38. Gandomi, A.; Alavi, A.; Shadmehri, D.M.; Sahab, M. An empirical model for shear capacity of RC deep beams using genetic-simulated annealing. *Arch. Civ. Mech. Eng.* **2013**, *13*, 354–369. [CrossRef]
39. Tran, D.H.; Luong, D.L.; Chou, J.S. Nature-inspired metaheuristic ensemble model for forecasting energy consumption in residential buildings. *Energy* **2020**, *191*, 116552. [CrossRef]
40. Bui, D.K.; Nguyen, T.; Chou, J.S.; Nguyen-Xuan, H.; Ngo, T.D. A modified firefly algorithm-artificial neural network expert system for predicting compressive and tensile strength of high-performance concrete. *Constr. Build. Mater.* **2018**, *180*, 320–333. [CrossRef]
41. Assegie, T.A.; Salau, A.O.; Badrudeen, T.U. Estimation of concrete compression using regression models. *Bull. Electr. Eng. Inform.* **2022**, *11*, 2799–2804. [CrossRef]
42. Wu, X.; Zhu, F.; Zhou, M.; Sabri, M.M.S.; Huang, J. Intelligent Design of Construction Materials: A Comparative Study of AI Approaches for Predicting the Strength of Concrete with Blast Furnace Slag. *Materials* **2022**, *15*, 4582. [CrossRef]
43. Ghunimat, D.; Alzoubi, A.E.; Alzboon, A.; Hanandeh, S. Prediction of concrete compressive strength with GGBFS and fly ash using multilayer perceptron algorithm, random forest regression and *k*-nearest neighbor regression. *Asian J. Civ. Eng.* **2022**, 1–9. [CrossRef]
44. Kumar, A.; Arora, H.C.; Kapoor, N.R.; Mohammed, M.A.; Kumar, K.; Majumdar, A.; Thinnukool, O. Compressive Strength Prediction of Lightweight Concrete: Machine Learning Models. *Sustainability* **2022**, *14*, 2404. [CrossRef]
45. Hussain, F.; Ali Khan, S.; Khushnood, R.A.; Hamza, A.; Rehman, F. Machine Learning-Based Predictive Modeling of Sustainable Lightweight Aggregate Concrete. *Sustainability* **2023**, *15*, 641. [CrossRef]
46. Azadkia, M. Optimal choice of *k* for *k*-nearest neighbor regression. *arXiv* **2020**, arXiv:1909.05495.
47. Migallón, V.; Navarro-González, F.J.; Penadés, H.; Penadés, J.; Villacampa, Y. A parallel methodology using radial basis functions versus machine learning approaches applied to environmental modelling. *J. Comput. Sci.* **2022**, *63*, 101817. [CrossRef]
48. Todeschini, R. k-nearest neighbour method: The influence of data transformations and metrics. *Chemom. Intell. Lab. Syst.* **1989**, *6*, 213–220. [CrossRef]
49. Afzal, S.; Wani, M.A. Comparative study of back propagation learning algorithms for neural networks. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2013**, *3*, 1151–1156.
50. Nwankpa, C.; Ijomah, W.; Gachagan, A.; Marshall, S. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *arXiv* **2018**, arXiv:1811.03378.
51. Aggarwal, C.C. *Neural Networks and Deep Learning*; Springer: Berlin/Heidelberg, Germany, 2018.
52. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
53. Liu, D.C.; Nocedal, J. On the limited memory BFGS method for large scale optimization. *Math. Program.* **1989**, *45*, 503–528. [CrossRef]
54. Rokach, L.; Maimon, O. *Data Mining with Decision Trees. Theory and Applications*; World Scientific: Singapore, 2007; Volume 69. [CrossRef]
55. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed.; Springer Series in Statistics; Springer: Berlin/Heidelberg, Germany, 2009. [CrossRef]
56. Hill, T.; Lewicki, P. *Statistics: Methods and Applications: A Comprehensive Reference for Science, Industry, and Data Mining*; StatSoft Inc.: Tulsa, OK, USA, 2006.
57. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
58. Biau, G.; Scornet, E. A random forest guided tour. *TEST* **2016**, *25*, 197–227. [CrossRef]
59. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
60. Friedman, J.H. Stochastic gradient boosting. *Comput. Stat. Data Anal.* **2002**, *38*, 367–378. [CrossRef]

61. Sutton, C.D. 11—Classification and Regression Trees, Bagging, and Boosting. In *Data Mining and Data Visualization*; Rao, C., Wegman, E., Solka, J., Eds.; Handbook of Statistics; Elsevier: Amsterdam, The Netherlands, 2005; Volume 24, pp. 303–329. [CrossRef]
62. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794. [CrossRef]
63. Rashmi, K.; Gilad-Bachrach, R. DART: Dropouts meet Multiple Additive Regression Trees. *arXiv* **2015**, arXiv:1505.01866v1.
64. Awad, M.; Khanna, R. *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*; Apress Media: New York, NY, USA, 2015. [CrossRef]
65. Zhang, F.; O'Donnell, L.J. Chapter 7—Support vector regression. In *Machine Learning*; Mechelli, A., Vieira, S., Eds.; Academic Press: Cambridge, MA, USA, 2020; pp. 123–140. [CrossRef]
66. Balasundaram, S.; Tanveer, M. On Lagrangian twin support vector regression. *Neural Comput. Appl.* **2013**, *22*, 257–267. [CrossRef]
67. Smola, A.J.; Schölkopf, B. A tutorial on support vector regression. *Stat. Comput.* **2004**, *14*, 199–222. [CrossRef]
68. Gani, W.; Taleb, H.; Limam, M. Support vector regression based residual control charts. *J. Appl. Stat.* **2010**, *37*, 309–324. [CrossRef]
69. Berrar, D. Cross-Validation. In *Encyclopedia of Bioinformatics and Computational Biology*; Ranganathan, S., Gribskov, M., Nakai, K., Schönbach, C., Eds.; Academic Press: Oxford, UK, 2019; pp. 542–545. [CrossRef]
70. Nguyen, H.; Bui, X.N. Predicting blast-induced air overpressure: A robust artificial intelligence system based on artificial neural networks and random forest. *Nat. Resour. Res.* **2019**, *28*, 893–907. [CrossRef]
71. Fernández-Fanjul, A.; Tenza-Abril, A.J. Méthode Fanjul: Dosage pondéral des bétons légers et lourds. *Ann. Bâtim. Trav. Publics* **2012**, *5*, 32–50.
72. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830. [CrossRef]
73. Ridgeway, G. Generalized Boosted Models: A Guide to the gbm Package, 2020. Available online: https://pbil.univ-lyon1.fr/CRAN/web/packages/gbm/vignettes/gbm.pdf (accessed on 10 January 2023).