



The impact of data splitting methods on machine learning models: A case study for predicting concrete workability

Hossein Babaei¹ · Mohammad Zamani¹ · Soheil Mohammadi¹

Received: 16 March 2025 / Accepted: 21 April 2025
© The Author(s), under exclusive licence to Springer Nature Switzerland AG 2025

Abstract

Artificial intelligence and machine learning are transforming the field of materials science by enabling predictive procedures for more efficient design. This may significantly affect traditional costly, time-consuming, and labor-intensive concrete evaluation test procedures, including the workability indicators. As a result, the available test data are limited, which in turn may introduce high variance and sampling effects that complicate the validity and comparability of research findings across studies. Concrete flow prediction, in particular, faces these challenges. Here, a well-known high-performance concrete dataset is adopted to predict the flow as the primary outcome, alongside slump and compressive strength, by using multi-output deep neural networks and applying the transfer learning technique. These approaches exploit the correlation between the mentioned properties, especially flow, to increase the predictive accuracy. The study demonstrates that robust flow predictions require advanced data-splitting techniques, specifically, the nested cross-validation technique. It differs from the k-fold cross-validation by including both test and validation sets with multiple folds for each, thereby minimizing the sampling effects and enhancing the prediction consistency. This rigorous framework addresses the high variance inherent in small datasets, a common limitation in engineering research. The findings highlight the importance of robust data-splitting strategies, which collectively reduce bias and improve reliability in material property prediction. This approach may pave the way for more sustainable, efficient and reliable evaluation procedures that could be applied in materials science and engineering disciplines at large.

Keywords Concrete Workability Prediction · Imbalance Engineering Dataset · ML · DNN · Data-splitting · Nested cross-validation

1 Introduction

Artificial intelligence (AI) includes computational technologies that mimic human intelligence, such as learning and problem solving [1]. A major part of AI is machine learning (ML), which involves implementing computer programs that learn from data and make predictions on their basis. ML can be divided into different methods, including supervised and unsupervised learning. Regression models are crucial for prediction tasks in supervised learning, aiming to interpolate values on the basis of the input data, helping with better decision-making and process optimization [2]. Deep

learning, a subset of ML, further improves these capabilities by using neural networks to discover complex patterns and relationships in data, a capability that can provide efficiency and innovation across different disciplines [3].

ML models have been instrumental in advancing scientific discoveries in various engineering fields, including electrical, mechanical, industrial, material, and civil engineering [4–11]. Traditional or more advanced ML techniques, for instance Physics Informed Neural Networks (PINNs) [12], offer significant potential for addressing difficult tasks in civil engineering by providing valuable insights into complex problems in various disciplines, including structural engineering, construction management, smart transportation, environmental science and concrete technology [13–17].

In construction and building materials, ML can play a significant role by predicting the mechanical properties of concrete on the basis of the basic constitutive ingredients

✉ Soheil Mohammadi
smoham@ut.ac.ir

¹ High-Performance Computing Laboratory, School of Civil Engineering, College of Engineering, University of Tehran, Tehran, Iran

of its mixture, leveraging the in-hand data from previous experimental tests to reduce the cost and time of further experimental tests significantly [18–21].

Concrete is one of the most commonly used materials in construction [22]. Traditional testing methods for measuring its mechanical properties, such as slump, flow and compressive strength, are crucial in assessing its workability and final strength [23, 24]. However, these methods are time-consuming and costly, and require labor and in situ equipment. In contrast, ML-aided testing offers a more efficient alternative, with a major environmental impact of reduction in CO₂ emissions [25–27] and lower costs [20, 25, 28].

Concrete workability plays a crucial role in ensuring the durability and strength of a structure, as it affects the ease with which concrete can be transported, placed, poured into molds, and compacted around reinforcement [29, 30]. Slump and flow tests are conventional methods for quantifying the concrete workability [31].

High-performance concrete (HPC) uses fly ash, ground granulated blast furnace slag and silica fume additions to enhance its performance. Typically, it has a lower water-to-cement ratio and smaller aggregate size, leading to greater strength and reduced permeability. These features make HPC suitable for conditions where higher durability and specific performance characteristics, such as an increased modulus of elasticity, are needed [32]. Clearly, developing a reliable model to determine or predict the properties of highly complex HPC materials is more challenging [33].

While many studies have focused on the properties of concrete, particularly compressive strength [34], Yeh [35] examined flow properties and compiled a dataset of seven input features for HPC mixing ingredients and output variables for slump, flow and compressive strength. The focus was on flow prediction via the combination of second-order regression and artificial neural network. Subsequently, Chen et al. [36] improved the estimation by developing a cubic gene-expression programming (GEP) with a parallel structure to overcome premature convergence and increase solution diversity. Later, they enhanced the prediction by adopting the Genetic Algorithm Combining Operation Tree (GAOT) to address the premature convergence, demonstrating better performance than regression models and Back Propagation Neural Networks (BPNNs) [37]. Moreover, Aydogmus et al. [33] investigated the use of bagging, a popular ensemble learning method, for constructing ensemble models. Moreover, Ünlü [38] used machine learning algorithms for predicting flow, highlighting methods that had not been previously used or compared in similar studies, such as the so-called ‘M5’ decision tree, with the best result obtained by the Multi-Layer Perceptron (MLP) method. More recently, Garcia et al. [26] employed metaheuristic machine learning techniques and deep learning to predict the slump and flow of concrete while achieving the best

performance in deep learning with ten neurons in a hidden layer. Yeh’s dataset has remained well studied among researchers because of its availability and multi-output basis, making it suitable for testing multi-output regression to predict slump, flow and compressive strength simultaneously [39–46].

Dataset splitting methods vary significantly in different studies, ranging from simple train–test splitting to more sophisticated cross-validation approaches. Garcia et al. [26] used 70% of the records for training, and the remaining records were used for validation, whereas Lee et al. [43] divided the dataset into 80 records for training and 23 records for validation. Emami and Martínez-Muñoz [46] applied the threefold cross-validation technique, whereas Yeh [35] and Schmid et al. [45] used 4- and fivefold cross-validation, respectively, and Aydogmus et al. [33] examined both threefold and fivefold cross-validation approaches. Additionally, several studies [38–41, 44] have utilized the tenfold cross-validation. Another less common method, the leave-one-out cross-validation, was employed by Chen and Ma [42]. Additionally, Chen et al. [36] used 94 records, dividing them into 50 for training, 20 for validation, and 24 for testing.

Failing to use separate validation and test sets may weaken the model evaluation by creating overlap in data usage, leading to overly optimistic results. A distinct test set provides an independent performance assessment, offering a clearer view of its real-world applicability. Notably, this issue may not occur with every dataset, although it occurs more frequently in small datasets [47]. Therefore, choosing an appropriate data-splitting method should depend on the nature of the data—for example, using PPSFL for sequential data [48].

Despite the use of various splitting methods, the validity and reliability of different methods remain uncertain to some extent [20]. In the early stage of the present study, the authors observed a high variance in model performance due to the sampling of flow data. In other words, the performance of the model varied significantly depending on the subset of records chosen for evaluation. Random selection of a portion of the dataset by a random seed, often led to inconsistent performance results with the same model and algorithm.

The present study delves into the exploration and comparison of various data splitting and model validation methods. The primary goal is to identify the most suitable method on the basis of the unique characteristics and challenges presented by the dataset in question. The authors place particular emphasis on the influence of random seed selection when applied to Yeh’s dataset across different data-splitting techniques, illustrating how this choice can significantly affect the model outcomes. This highlights the critical importance of treating the validation and test sets as

separate entities. Instead of relying on a single set for validation and a single set for testing, the authors advocate for the use of k-fold cross-validation not only during the validation phase but also during the testing phase. By using the nested cross-validation method on Yeh's dataset, the study effectively addresses the discussed issues. This method leads to more consistent and reliable results, thereby improving the comparability of outcomes between different studies and reducing discrepancies. By trying to fill the gap in addressing random seed sensitivity and data-splitting challenges, especially in small datasets, through robust validation, this study further employs a diverse array of ML models via the proposed splitting method. Furthermore, a diverse array of ML models is employed via the proposed splitting method. These range from traditional machine learning algorithms to various deep learning approaches. This study explores both single-output and multi-output deep learning frameworks and incorporates techniques such as the transfer learning to increase model performance.

The main features of the present work are highlighted as:

- (i) A structured evaluation framework for assessing concrete workability in a more efficient and sustainable manner.
- (ii) Consistency in data splitting to ensure reliable and reproducible model assessment.
- (iii) Nested cross-validation to minimize sampling effects and improve evaluation robustness.
- (iv) Deep neural networks (DNNs) for enhancing accuracy in predicting concrete properties from imbalanced datasets.
- (v) Multi-output DNNs and transfer learning to exploit shared property correlations for better flow prediction.

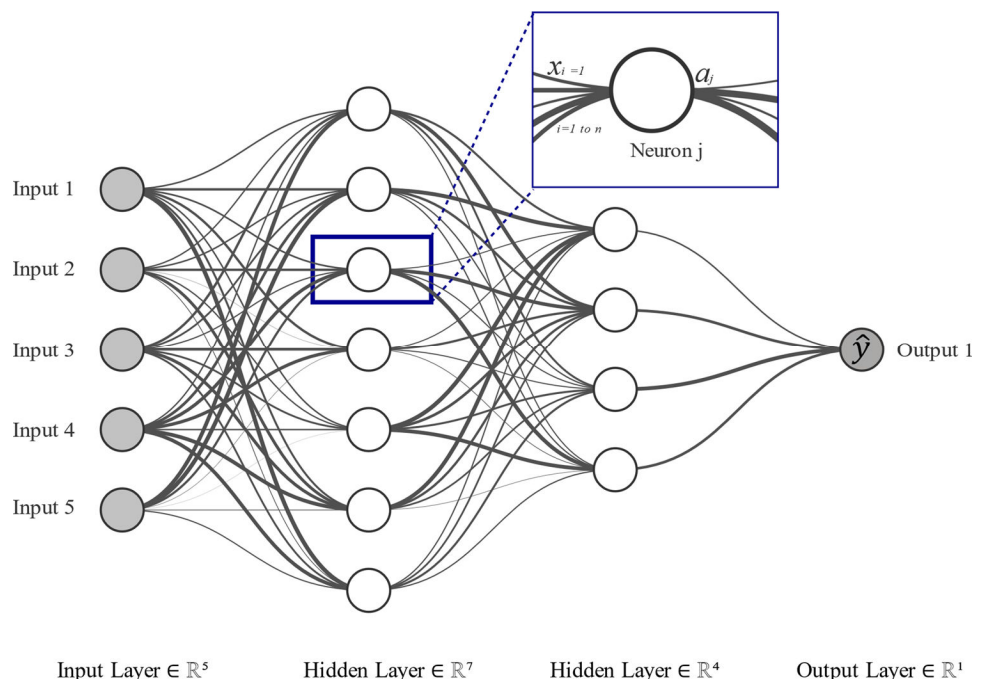
The remainder of the paper is organized as follows. First, Sect. 2 provides a brief overview of deep learning and its various approaches, including single- and multi-output deep neural networks (DNNs) and transfer learning techniques. Building upon these foundations, Sect. 3 discusses different methods of data splitting techniques and presents algorithms for their implementation. Subsequently, Sect. 4 describes the characteristics and preprocessing of the dataset used in this case study, followed by Sect. 5, which details the experimental setup and model architectures. Then, Sect. 6 presents the results along with a comprehensive discussion on the existing implications, which is followed by Sect. 7, which concludes the paper and suggests potential directions for future research.

2 Deep Neural Network (DNN) Regressors

A deep neural network (DNN) regressor [49, 50] is a specific machine learning model for predicting numerical values. Its architecture consists of an input layer, multiple hidden layers, and an output layer (Fig. 1). Each layer contains a set of artificial neurons interconnected by weighted edges, facilitating the flow of information through the network.

As DNNs are widely known and extensively documented, only the main necessary descriptions are provided.

Fig. 1 A sample deep neural network with an input layer ($n \in \mathbb{R}^5$), two hidden layers with seven and four neurons, and an output layer ($n \in \mathbb{R}^1$). The zoomed window on a hidden neuron shows the weighted inputs and the activation output a_j . Darker signal lines indicate that the corresponding output signal from layer $j-1$ is associated with a higher weight in the connected neuron of the next layer (j). The figure was generated via [51] and further processed for clarity and detail



2.1 Training process

In a fully connected neural network, each neuron j receives n (number of previous neurons) input signals (x_i) from the neurons in the previous layer. The neuron computes a weighted sum of its inputs:

$$z_j = \sum_{i=1}^n w_{ij}x_i + b_j, \quad (1)$$

where w_{ij} represents the weight between neuron i in the previous layer and neuron j in the current layer and b_j is the bias term associated with neuron j .

The weighted sum z_j is then passed through an activation function $\phi(z_j)$ to introduce nonlinearity into the model.

$$a_j = \phi(z_j) \quad (2)$$

This process of passing the output of each neuron a_j to the next layer continues until the final output layer is reached. For a regression task, the output layer typically consists of a single neuron that produces the predicted value $\hat{y}$.

The difference between the predicted value $\hat{y}$ and the actual target value y is quantified by the loss function $L(\hat{y}, y)$. A commonly used loss function in regression tasks is the mean squared error (MSE):

$$L(\hat{y}, y) = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i - y_i)^2, \quad (3)$$

where m is the number of samples.

To minimize this loss, model parameters (weights w_{ij} and biases b_j) are optimized through the backpropagation process, in which gradients of the loss function with respect to the weights and biases are computed.

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_j} \cdot \frac{\partial a_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ij}} \quad (4)$$

$$\frac{\partial L}{\partial b_j} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_j} \cdot \frac{\partial a_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial b_j} \quad (5)$$

These gradients are then used to update the weights and biases. One common method for updating the parameters is the gradient descent method, where the weights w_{ij} and biases b_j are updated in the direction that minimizes the loss function

$$w_{ij} \leftarrow w_{ij} - \eta \cdot \frac{\partial L}{\partial w_{ij}}, \quad (6)$$

$$b_j \leftarrow b_j - \eta \cdot \frac{\partial L}{\partial b_j}, \quad (7)$$

where η is the learning rate hyperparameter for controlling the step size during the gradient descent process. The gradient descent update algorithm continues for a predefined number of epochs to reach convergence or to meet an early stopping criterion [52]. Notably, other alternative approaches, such as adaptive momentum and RMSProp, may enhance convergence and/or performance in specific DNN problems [53].

2.2 Hyperparameter tuning

Hyperparameter tuning [54] plays a critical role in optimizing regression models, as it significantly impacts their ability to generalize effectively to new unseen data. Unlike model parameters, which are learned during the training phase, hyperparameters are predefined settings that influence various aspects of the learning process. They control factors such as model complexity, architecture, learning rate, and regularization strength, among others. Proper tuning of these hyperparameters is essential to balance underfitting and overfitting [55], ensuring that the model performs robustly across different datasets [56]. This tuning process typically involves the systematic exploration of various combinations of hyperparameters. A common approach to achieve this is the cross-validation concept, where the data are split into multiple folds and the model is trained and validated across these folds for each hyperparameter combination. Techniques such as grid search, random search or more advanced methods such as the Bayesian optimization are often employed with cross-validation to identify the optimal set of hyperparameters for enhanced overall performance [54, 56].

2.3 Multi-Output-DNN

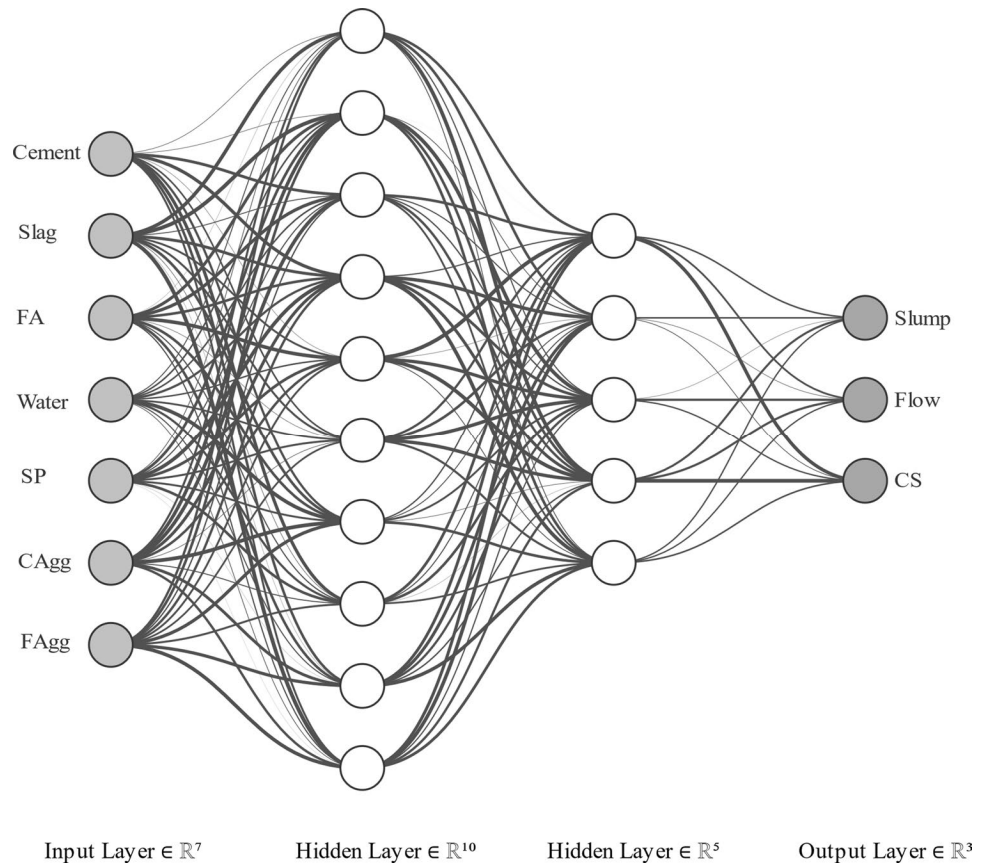
A Multi-Output-DNN (MODNN) is designed to predict multiple outputs simultaneously. Unlike a single-output-DNN (SODNN), the output layer contains various neurons, each corresponding to a different predicted variable. The network processes inputs through its hidden layers, and each neuron in the output layer produces a separate output (Fig. 2).

The loss function in MODNN typically sums the individual losses of each output. For example, if MSE is used for each output, the total loss L can be defined as

$$L(\hat{y}, y) = \frac{1}{k \cdot m} \sum_{i=1}^m \sum_{o=1}^k (\hat{y}_{io} - y_{io})^2, \quad (8)$$

where k is the number of outputs the model is supposed to predict, $\hat{y}_{io}$ is the predicted value of the o^{th} output of the i^{th} sample, and y_{io} is the actual value of the o^{th} output of the i^{th} sample.

Fig. 2 A sample multi-output deep neural network with an input layer ($n \in \mathbb{R}^7$), two hidden layers with ten and five neurons, and an output layer ($n \in \mathbb{R}^3$) (the input and output layers are based on the case study in Sect. 4)



Backpropagation adjusts the weights to minimize this combined loss to optimize the network performance across all outputs:

$$\frac{\partial L}{\partial w_{ij}} = \sum_{o=1}^m \frac{\partial L}{\partial \hat{y}_o} \cdot \frac{\partial \hat{y}_o}{\partial a_{j,o}} \cdot \frac{\partial a_{j,o}}{\partial z_{j,o}} \cdot \frac{\partial z_{j,o}}{\partial w_{ij}}, \quad (9)$$

$$\frac{\partial L}{\partial b_j} = \sum_{o=1}^m \frac{\partial L}{\partial \hat{y}_o} \cdot \frac{\partial \hat{y}_o}{\partial a_{j,o}} \cdot \frac{\partial a_{j,o}}{\partial z_{j,o}} \cdot \frac{\partial z_{j,o}}{\partial b_j}, \quad (10)$$

where all parameters are consistent with those defined in (4) and (5). Note that the summation is performed separately for each output, as indicated by the index o .

The weights can then be updated via an optimization method such as the gradient descent (6) and (7). The MODNN architecture is useful in scenarios where outputs are related or when predicting multiple variables in a single model can improve efficiency and accuracy [46].

2.4 Transfer learning

Initialization of weights and biases in a neural network is critical to efficient and accurate training [57]. Common

approaches include random initialization and more innovative techniques such as Xavier initialization and He initialization [58]. In some cases, the transfer learning technique is applied, where the initial weights and biases of the model are derived from the trained parameters of another model performing a similar task [59]. These methods collectively attempt to ensure that the network starts from a well-informed point to facilitate efficient learning with the goal of improved performance.

2.5 Evaluation metrics

Evaluation metrics in regression tasks assess how well a model predicts outcomes by comparing predicted and actual values. Evaluation metrics provide quantitative measures of prediction accuracy and help determine the performance of a model in capturing the underlying data, guiding the optimization process, and ensuring reliable and accurate predictions in real-world applications.

RMSE (root mean square error) and MAE (mean absolute error) are defined as:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}, \quad (11)$$

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|, \quad (12)$$

are commonly used metrics for assessing model performance by quantifying prediction errors. RMSE calculates the square root of the average squared differences between the predicted and observed values, whereas MAE computes the average of the absolute differences. Both metrics range from zero to infinity, with lower values indicating better performance. Furthermore, the coefficient of determination (R^2)

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}, \quad (13)$$

measures the precision of fit for a regression model, generally ranging up to 1, with higher values representing a better fit, although it can be negative if the model performs poorly [60].

Table 1 Definition of the adopted splitting methods

Splitting Method	Description
SM1V	Train-Validation Splitting
SM1T	Train-Test Splitting
SM2	Train-Validation-Test Splitting
SM3	K-fold Cross-Validation Without Test Set
SM4	K-fold Cross-Validation With Test Set
SM5	Nested Cross-Validation

3 Data splitting

In all data splitting methods, there is always a subset called the training set. Some approaches also include both a validation set and a test set, whereas others may include only one. The key difference between the validation and test sets is that the validation set is typically used during model development to fine-tune hyperparameters and adjust the model architecture. In contrast, the test set is reserved for evaluating the final performance of the model without involving any training and fine-tuning procedures. The validation set allows the model to be evaluated multiple times without compromising the integrity of the final performance measure. In contrast, the test set provides an unbiased estimate of the model performance after the training and validation phases are completed.

This section describes various methods of data splitting [61, 62], as briefly defined in Table 1.

3.1 SM1V

SM1V divides the dataset into training and validation sets. The training set is used to train the model, whereas the validation set is used to tune the model hyperparameters and evaluate its performance without the use of a separate test set for the final assessment (Box 1). The random seed indicates which subset of records of the dataset are assigned to the validation sets (Fig. 3).

1. Training Phase:

1.1. Shuffle the entire dataset.

1.2. Split the dataset into two sets:

1.2.1. Train set (e.g., 80% of the data)

1.2.2. Validation set (e.g., 20% of the data)

1.3. Repeat for t sets of hyperparameters:

1.3.1. Train the model using the training set.

1.3.2. Evaluate the model using the validation set.

2. Assessment Phase:

2.1. Choose the best set of hyperparameters based on the performance of the model on the validation set using the chosen evaluation metric.

2.2. Report the performance of the chosen model and hyperparameters as the final assessment on the validation set.

Box 1 SM1V Algorithm

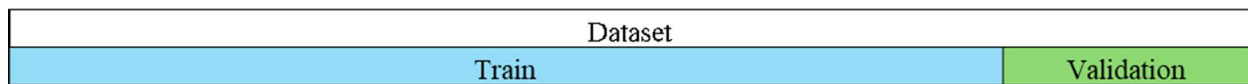


Fig. 3 Visualized splitting method SM1V

1. Training Phase:

1.1. Shuffle the entire dataset.

1.2. Split the dataset into two sets:

1.2.1. Train set (e.g., 80% of the data)

1.2.2. Test set (e.g., 20% of the data)

1.3. Repeat for t sets of hyperparameters:

1.3.1. Train the model using the training set.

1.3.2. Evaluate the model using the training set.

2. Assessment Phase:

2.1. Choose the best set of hyperparameters based on the performance of the model on the training set using the chosen evaluation metric.

2.2. Report the performance of the chosen model and hyperparameters as the final assessment on the test set.

Box 2 SM1T Algorithm

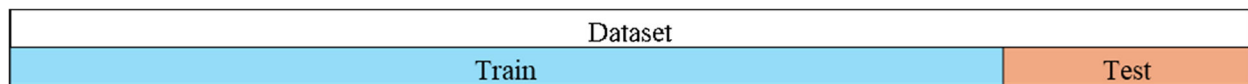


Fig. 4 The splitting method SM1T

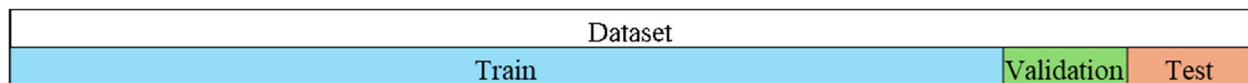


Fig. 5 Visualized splitting method SM2

3.2 SM1T

SM1T (Box 2) divides the dataset into training and test sets. The training set is used to both train the model and tune the model hyperparameters, whereas the test set is employed to evaluate its performance. Unlike the SM1V approach, SM1T uses a dedicated test set for the final assessment, providing an independent evaluation of the model. The random seed determines which subset of the dataset is assigned to the test set (Fig. 4).

3.3 SM2

SM2 divides the dataset into training, validation, and test sets (Fig. 5). The training set is used to train the model, the validation set is used for hyperparameter tuning and model selection [63], and the test set is reserved for the final assessment of model performance (Box 3). The random seed determines which records from the dataset are included in the test set. This means that the final assessment report is based only on the portion of the dataset selected by chance, which may not accurately represent the overall performance of the model on the entire dataset.

1. Training Phase:*1.1. Shuffle the entire dataset.**1.2. Split the dataset into three sets:**1.2.1. Training set (e.g., 80% of the data)**1.2.2. Validation set (e.g., 10% of the data)**1.2.3. Test set (e.g., 10% of the data)**1.3. Repeat for t sets of hyperparameters:**1.3.1. Train the model using the training set.**1.3.2. Evaluate the model using the validation set.***2. Assessment Phase:***2.1. Choose the best set of hyperparameters based on the performance of the model on the validation set using the chosen evaluation metric.**2.2. Evaluate the chosen model on the test set.**2.3. Report the performance of the chosen model and hyperparameters as the final assessment on the test set.***Box 3** SM2 Algorithm**3.4 SM3**

SM3 (Box 4) involves splitting the dataset into k subsets or folds. For the k iteration, $k - 1$ folds are used to train the model, and the remaining fold is used for validation. This process is repeated k times, with each fold being used exactly once as the validation set (Fig. 6). The performance metric is averaged over the k validation sets to evaluate the model comprehensively. This method does not include a separate test set for the final model assessment.

In SM3, each data point is considered to be in the validation set exactly once. The random seed controls which specific records are selected for the validation set in each iteration.

3.5 SM4

SM4 (Box 5) extends the previous method by reserving a portion of the dataset as a test set. The remaining data records (outer-train) are split into k folds, as in the SM3

1. Training Phase:*1.1. Shuffle the entire dataset.**1.2. Split the dataset into k folds.**1.3. Repeat for t sets of hyperparameters:**1.3.1. For each iteration ($i = 1$ to k):**1.3.1.1. Use the i^{th} fold as the validation set.**1.3.1.2. Use the remaining $k-1$ folds as the training set.**1.3.1.3. Train the model using the training set.**1.3.1.4. Evaluate the model using the validation set.**1.3.2. Calculate the average performance metric across all k folds.***2. Assessment Phase:***2.1. Choose the best set of hyperparameters based on the averaged performance of the model on all k folds using the chosen evaluation metric.**2.2. Report the performance of the chosen model and hyperparameters as the final assessment.***Box 4** SM3 algorithm

	Dataset		
Iteration=1	Train		Validation
Iteration=2	Train		Validation
Iteration=3	Train		Validation
Iteration=4	Train	Validation	Train
Iteration=5	Validation	Train	

Fig. 6 Visualized splitting method SM3 ($k = 5$)

1. Training Phase:

1.1. Shuffle the entire dataset.

1.2. Split the dataset into two sets:

1.2.1. Outer-train set (e.g., 85% of the data)

1.2.2. Test set (e.g., 15% of the data)

1.3. Split the outer-train set into k folds.

1.4. Repeat for t sets of hyperparameters:

1.4.1. For each inner iteration ($i = 1$ to k):

1.4.1.1. Use the i^{th} fold as the validation set.

1.4.1.2. Use the remaining $k-1$ folds as the training set.

1.4.1.3. Train the model using the training set.

1.4.1.4. Evaluate the model using the validation set.

1.4.2. Calculate the average performance metric across all k folds.

2. Assessment Phase:

2.1. Choose the best set of hyperparameters based on the averaged performance of the model on the whole k -folds using the chosen evaluation metric.

2.2. Evaluate the chosen model on the test set.

2.3. Report the performance of the chosen model and hyperparameters as the final assessment on the test set.

Box 5 SM4 algorithm

method. The model is trained and validated via the same k -fold cross-validation procedure as in SM3, with the best-performing model being evaluated on the reserved test set. This method combines the advantages of k -fold cross-validation (SM3) for model selection with the robustness of using a separate test set for the final performance evaluation. Similar to SM2, the random seed determines which records are allocated to the test set by chance, and the final performance report is based on a randomly selected test set, which may not fully represent the overall performance of the model across the entire dataset (Fig. 7).

3.6 SM5

Nested cross-validation [61] is a robust technique that addresses potential bias in hyperparameter tuning and model

selection. It involves two layers of cross-validation: an outer iteration and an inner iteration (Fig. 8). In the outer iteration, the entire dataset is divided into k folds, with each fold taking turns as the test set, while the remaining $k-1$ folds form the outer-training set. For each outer iteration, the inner iteration performs k' -fold cross-validation on the outer-training set to tune the hyperparameters and select the best model configuration. This process results in k independently trained and optimized models, each evaluated on its corresponding reserved test fold. The final assessment is obtained by averaging the results across all k -test folds (Box 6). This approach ensures that each record in the dataset is used as a test case exactly once, with the final evaluation being the average performance across the entire dataset. In nested cross-validation, the choice of random seed has a minimal impact because every data point will eventually appear in

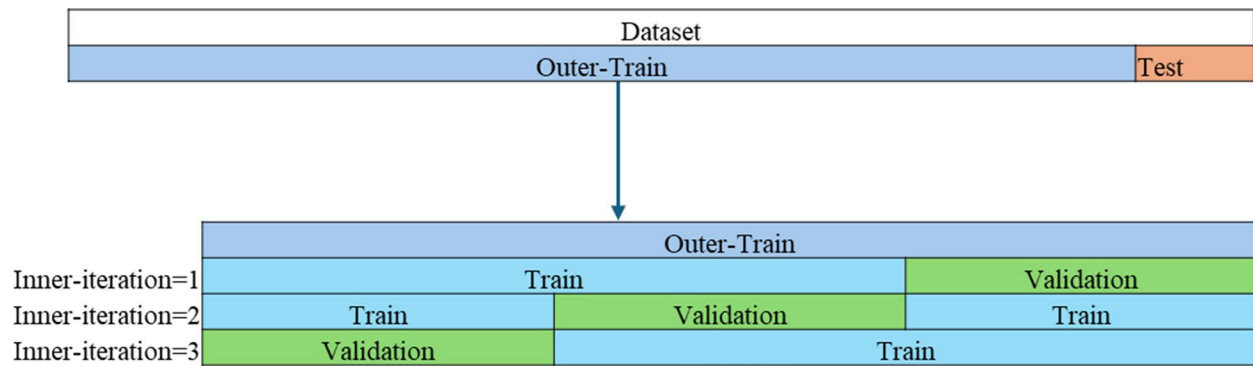


Fig. 7 Visualized splitting method SM4 ($k=3$)

the test set exactly once. The random seed influences only the assignment of data points to specific folds during the process. Since all the data points are evaluated in their own test set, the significance of the random seed decreases, making it less critical to the overall results and final assessment.

4 Dataset characterization

In this study, Yeh's dataset, which comprises 103 records, each representing a unique mix of HPC, is adopted. The output variables are slump, flow, and 28-day compressive strength (CS). The input features are cement (Cement), slag (Slag), fly ash (FA), water (Water), superplasticizer (SP), coarse aggregate (CAgg), and fine aggregate (FAgg). All material properties in the dataset are consistent with the same environmental and procedural conditions during the mix preparation. Given the small size of the dataset, no outliers were identified or removed, as eliminating any data

points could have negatively affected the analysis. For pre-processing, data standardization was applied to all training sets, with the test and validation sets transformed accordingly. The statistical summaries of the input and output variables are presented in Table 2 and Table 3, respectively.

There are various techniques to understand the relationship between input and output features, such as performing a sensitivity analysis [64] after model training and employing a Pearson correlation matrix [65] before training the model. In Table 4, the Pearson correlation matrix can be observed.

The Pearson correlation matrix reveals several key insights into the relationships between concrete mix components and properties. As expected, slump and flow, both indicators of concrete workability, are strongly positively correlated (0.91). Interestingly, despite the usual expectation that the superplasticizer would improve the workability, it has a negative correlation with both slump (-0.21) and flow (-0.18), suggesting a more complex interaction. Water, as expected, positively correlates with slump (0.47) and flow

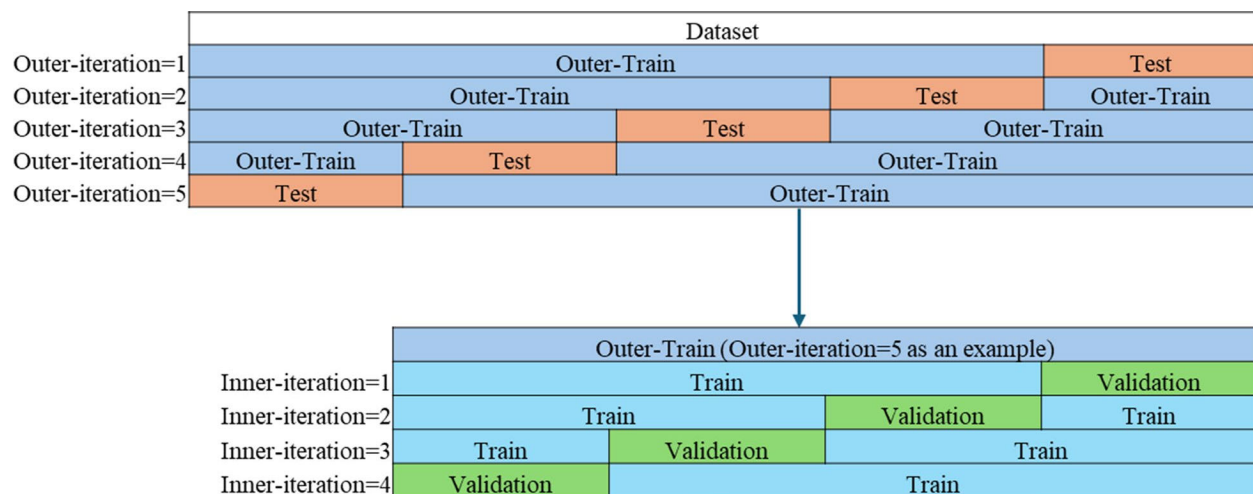


Fig. 8 Visualized splitting method SM5 (Nested cross-validation) ($k'=5$, $k=4$)

1. *Shuffle the entire dataset.*
2. *Split the dataset into k ' outer folds.*
3. *For each outer iteration ($i = 1$ to k):*
 - 3.1. *Split the Outer-Train set into k folds.*
 - 3.2. *Repeat for t sets of hyperparameters:*
 - 3.2.1. *For each inner iteration ($j = 1$ to k):*
 - 3.2.1.1. *Use the j^{th} fold as the validation set.*
 - 3.2.1.2. *Use the remaining $k-1$ folds as the training set.*
 - 3.2.1.3. *Train the model using the training set.*
 - 3.2.1.4. *Evaluate the model using the validation set.*
 - 3.2.2. *Calculate the average performance metric across all k folds.*
 - 3.3. *Choose the best set of hyperparameters based on the averaged performance of the model on the whole k folds using the chosen evaluation metric.*
 - 3.4. *Evaluate the final i^{th} model on the i^{th} test set.*
4. *Report the average of all k '-Test set performance as the final assessment.*

Box 6 SM5 (Nested cross-validation) Algorithm**Table 2** Statistical summaries of the inputs

	Cement (kg/m3)	Slag (kg/m3)	FA (kg/m3)	Water (kg/m3)	SP (kg/m3)	CAGg (kg/m3)	FAGg (kg/m3)
Min	137	0	0	160	4.40	708	640.60
Max	374	193	260	240	19	1049.90	902
Mean	229.89	77.97	149.01	197.16	8.54	883.98	739.61
Standard deviation	78.87	60.46	85.41	20.20	2.80	88.39	63.34

Table 3 Statistical summaries of outputs

	Slump (cm)	Flow (cm)	CS (MPa)
Min	0	20	17.19
Max	29	78	58.53
Mean	18.05	49.61	36.04
Standard deviation	8.75	17.57	7.84

(0.63) due to its direct impact on workability but negatively correlates with compressive strength (CS) (-0.25), reflecting the well-known inverse relationship. Cement is positively correlated with both slump (0.15) and flow (0.19), as well as with CS (0.45), which matches the expected role of cement in concrete strengthening. While these correlations highlight expected trends, they highlight interesting aspects of component interactions, indicating that this dataset would be quite complex to analyze thoroughly. A sample of the

multi-output deep neural network, which is based on the dataset, is shown in Fig. 2.

5 Model setup

5.1 Hyperparameter tuning

This study utilizes the Optuna library as the main tool for hyperparameter optimization. Optuna leverages various algorithms, including the tree-structured Parzen estimator (TPE) sampler and grid search, among others [66]. Each hyperparameter is evaluated on the basis of the adopted data-splitting method.

5.2 SODNN & MODNN

DNNs are implemented in PyTorch [67]. To effectively prevent overfitting due to dataset complexity and limited sample size, several key techniques are incorporated: batch

Table 4 The Pearson correlation matrix of the dataset

	Cement	Slag	FA	Water	SP	CAgg	FAgg	Slump	Flow	CS
Cement	1	-0.24	-0.49	0.22	-0.11	-0.31	0.06	0.15	0.19	0.45
Slag	-0.24	1	-0.32	-0.03	0.31	-0.22	-0.18	-0.28	-0.33	-0.33
FA	-0.49	-0.32	1	-0.24	-0.14	0.17	-0.28	-0.12	-0.06	0.44
Water	0.22	-0.03	-0.24	1	-0.16	-0.6	0.11	0.47	0.63	-0.25
SP	-0.11	0.31	-0.14	-0.16	1	-0.1	0.06	-0.21	-0.18	-0.04
CAgg	-0.31	-0.22	0.17	-0.6	-0.1	1	-0.49	-0.19	-0.33	-0.16
FAgg	0.06	-0.18	-0.28	0.11	0.06	-0.49	1	0.2	0.19	-0.15
Slump	0.15	-0.28	-0.12	0.47	-0.21	-0.19	0.2	1	0.91	-0.22
Flow	0.19	-0.33	-0.06	0.63	-0.18	-0.33	0.19	0.91	1	-0.12
CS	0.45	-0.33	0.44	-0.25	-0.04	-0.16	-0.15	-0.22	-0.12	1

Table 5 Tuning ranges of hyperparameters

Hyperparameters	Range
Number of hidden layers	1–3 (integer)
Number of neurons in each hidden layer	1–128 (integer)
Activation function	[relu, leaky relu, tanh, sigmoid]
Learning rate	0.000001–0.01 (float)
Batch size	[not using minibatch training, 18, 36]
Weight decay rate (regularization parameter)	0.000001–1 (float)
Dropout rate	[not using dropout, 0.2, 0.5, 0.8]
Optimizer	[Adam, SGD with momentum = 0.8]

normalization [68], mini-batch training, dropout [69] and L2 regularization [70]. The training process is further optimized via two distinct algorithms: Adam [71] and SGD with momentum. Four activation functions (ReLU, Leaky ReLU, Sigmoid and tanh) [72] are examined to achieve the best efficiency. Early stopping [52] is employed as an additional measure against overfitting. The training is set to stop if the mean squared error (MSE) on the training set does not decrease by at least one unit over 50 epochs, allowing a maximum of 3000 epochs. The network architecture is carefully optimized through rigorous hyperparameter tuning, focusing on the number of layers and neurons per layer and all other tunable parameters detailed in Table 5.

These hyperparameters are used for all DNN models. Three approaches are evaluated for MODNN models. The first approach includes all three outputs (slump, flow, and compressive strengths) as targets (MODNN-AT). The second approach targets only slump and flow (MODNN-FS), whereas the third approach focuses on flow and compressive strength (MODNN-FC).

5.3 Transfer learning

The learned weights and biases from the models trained on slump and compressive strength are transferred to initialize the SODNN-Flow. The process begins by training

two independent SODNNs on the slump (SODNN-Slump) and compressive strength (SODNN-CS) as source models. Hyperparameter tuning is conducted for both source models to achieve optimal performance. After the best-performing model is obtained, the weights and biases of the source model are transferred to the initial layers of the flow prediction models, followed by further hyperparameter tuning.

Two distinct approaches to transfer learning are explored in this study, namely TL1 and TL2, which differ in how the data splits are handled during the transfer process.

In the first approach (TL1), a strict consistency is maintained between the data partitions of the source and target models. Specifically, if a particular record is assigned to the test set during the training of the source models (SODNN-Slump or SODNN-CS), that same record must also remain in the test set when training the flow model (SODNN-Flow). This constraint ensures that there is no information leakage between the source and target tasks, thereby keeping the evaluation fair and unbiased. In other words, the flow model is never allowed to see the input features of any record in its training phase if that record was used in the test set during source model training.

In the second approach (TL2), the above constraint is not enforced. A record that was part of the test set during the training of the source models may appear in the training set when training the flow model. While this may seem harmless at first, it introduces a risk of information leakage,

Table 6 Abbreviations for DNN models and transfer learning approaches

Abbreviate	Explanation
SODNN-Slump	SODNN model predicting Slump
SODNN-CS	SODNN model predicting CS
SODNN-Flow	SODNN model predicting Flow
TLS1-Flow	SODNN model predicting Flow with initial weights and biases of SODNN-Slump based on the first approach of transfer learning
TLC1-Flow	SODNN model predicting Flow with initial weights and biases of SODNN-CS based on the first approach of transfer learning
TLS2-Flow	SODNN model predicting Flow with initial weights and biases of SODNN-Slump based on the second approach of transfer learning

Table 7 Abbreviations of the splitting methods (SM1–SM5)

Splitting method abbreviations	Description
SM1V-v	SM1V method with v records allocated for the validation set
SM1T-t	SM1T method with t records allocated for the test set
SM2-v-t	SM2 method with v records for validation and t records allocated for testing
SM3-k	SM3 method utilizing k-fold cross-validation
SM4-k-t	SM4 method with t records allocated for testing and k'-fold cross-validation for training
SM5-k'-k	SM5 method utilizing k' outer folds and k inner folds

Table 8 Randomly selected seeds (RS1–RS6) used for all the experiments

Random seed	scenario
15	RS1
772	RS2
568	RS3
684	RS4
74	RS5
42	RS6

as the flow model could indirectly benefit from exposure to input features that were part of the test set in the source tasks. Notably, initial results based on TL2 overlooked such a difference without realizing the importance of the data splitting process.

It is important to acknowledge that it is not always evident that how small differences in the way data is divided can impact the results, especially when using transfer learning with a single dataset and multiple target variables Table 6.

6 Results and discussion

For each adopted splitting method, described in Table 7, six seeds are randomly selected from the range of 0–1000, as defined in Table 8. These seeds, labeled RS1 through RS6,

are consistently used for all the experiments. The R^2 evaluation metric is used as the primary focus for performance assessment.

6.1 Performance assessment of splitting methods

6.1.1 Linear regression on SM1T and one thousand random seeds

A simple linear regression [73] model is employed via an SM1T-10 approach using 1000 random seeds ranging from 0–1000. The resulting RMSE loss on the test set varies between 8.66 and 19.44, highlighting the significant impact of the choice of evaluation data (by changing the random seed) on the same model performance.

6.1.2 Comparison of SM1T and SM1V via the H2O library

The H2O library, a versatile Python tool, is utilized to streamline the evaluation of datasets by testing a variety of machine learning models [74]. The implementation included classical models such as XGBoost, gradient boosting machines, random forests and generalized linear models.

Table 9 and Table 10 display the performance of the H2O library, evaluated once on SM1V-10 and once on SM1T-10, respectively, by testing one hundred hyperparameters across H2O models for each RS in both splitting methods.

Table 9 Performance of the H2O library on SM1T-10

Random Seed	Train		Test		
	RMSE	MAE	RMSE	MAE	R2
RS1	3.68	2.22	13.92	10.15	0.46
RS2	7.15	4.93	12.30	11.05	0.67
RS3	13.53	11.61	17.52	15.98	0.28
RS4	8.96	7.13	15.54	14.44	0.34
RS5	9.79	8.02	19.06	13.54	-0.57
RS6	7.50	5.39	15.51	13.76	-0.79
Max – Min	9.85	9.39	6.77	5.84	1.46
Average	8.43	6.55	15.64	13.15	0.06
Standard deviation	2.98	2.91	2.21	1.99	0.54

Table 10 Performance of the H2O library on SM1V-10

Random Seed	Train		Validation		
	RMSE	MAE	RMSE	MAE	R2
RS1	1.42	1.09	11.48	7.89	0.63
RS2	2.66	2.00	8.13	7.05	0.86
RS3	0.42	0.31	12.03	9.83	0.66
RS4	1.60	1.24	8.77	7.36	0.79
RS5	5.90	4.44	12.06	9.42	0.37
RS6	14.80	12.85	10.45	8.71	0.19
Max – Min	14.38	12.53	3.93	2.77	0.67
Average	4.47	3.66	10.48	8.38	0.58
Standard deviation	4.93	4.31	1.55	1.03	0.23

Comparing Table 9 and Table 10 reveals a significant performance difference when the same records are used for validation ($0.58 R^2$) and testing ($0.06 R^2$). In SM1V, the validation set is used during the model tuning and evaluation, influencing the final model. In contrast, SM1T uses only the test set to evaluate the final model, with no role in training.

Additionally, the variance across different random seeds suggests that changing the records in the test or validation sets can lead to substantial differences in performance when the same algorithm is used.

6.1.3 Comparing the SODNN-Flow and H2O libraries on SM2

Having observed a notable difference between the validation and test sets, the SODNN-Flow and H2O models on SM2-18–10 are now compared, incorporating both the validation and test sets. The comparison is conducted using the random seeds RS1 through RS6 for both models to ensure consistent evaluation across different sets. SODNN-Flow hyperparameter tuning is conducted separately for each scenario, with 100 hyperparameter sets ($t = 100$), similar to H2O, and 100 configurations are tested across its models. Table 11 and Table 12 present the performance results of both models on SM2-18–10.

The comparison between SODNN-Flow and the H2O library on the SM2-18–10 dataset reveals a slight advantage for SODNN-Flow, with an average R^2 of 0.14 versus 0.10 for H2O on the test set. However, this difference is not substantial enough to conclusively determine which model performs better, justifying further evaluation of these models in Sect. 6.1.8. Moreover, the performance of both models suggests that relying on a single validation set may be insufficient to achieve reliable predictive accuracy.

A high variance is observed in the results solely because of changes in the random seed used for the data splitting method. Additionally, although the average result is clearly

unsatisfactory, certain random seeds exhibit notably high performance. This fluctuation complicates the evaluation of the model and makes it challenging to compare different studies.

6.1.4 SODNN-Flow and SODNN-CS on SM2

The SODNN is selected to assess whether this effect remains consistent when the compressive strength in this dataset is predicted. Six different scenarios are examined by varying the random seed used for SM2-18–10. Table 13 indicates that the results for CS are less sensitive to the choice of random seed. The study concludes that the high variance caused by sampling and changing the random seed is not necessarily observed in every task, even when a single dataset is used, and it should be examined carefully and checked in each context.

6.1.5 Comparing SM3 and SM4 via SODNN-Flow

According to the previous sections, the performances of the examined splitting methods are not sufficiently acceptable. Therefore, more complex splitting methods involving k-fold cross-validation are now being examined. SM3-10 and SM4-5–10 are examined via random seeds RS1 through RS6 for both models, and hyperparameter tuning is conducted one hundred times per RS. The performance results of SODNN-Flow on SM3-10 and SM4-5–10 are presented in Table 14 and Table 15, respectively.

Comparing Table 14 and Table 15 clearly reveals that the primary difference between SM3 and SM4 lies in the reserved test set in SM4. This difference results in a notable gap between the validation performance of SM3 ($R^2 = 0.69$) and the test set performance of SM4 ($R^2 = 0.28$). The significant difference between the validation and test set results highlights that relying solely on SM3 may not provide a reliable representation of the model performance on unseen future data.

Table 11 Performance of the H2O library on SM2-18–10

Random Seed	Train		Validation		Test		
	RMSE	MAE	RMSE	MAE	RMSE	MAE	R2
RS1	0.55	0.29	9.78	6.97	14.82	8.96	0.38
RS2	10.09	8.30	9.69	7.94	11.05	9.86	0.74
RS3	10.26	8.42	8.30	6.94	16.87	13.64	0.33
RS4	7.01	5.16	11.41	10.18	15.50	14.61	0.34
RS5	3.32	2.34	9.54	7.71	15.14	11.49	0.01
RS6	8.07	6.30	12.06	10.71	17.20	15.39	−1.20
Max – Min	9.71	8.12	3.76	3.77	6.15	6.43	1.94
Average	6.55	5.13	10.13	8.41	15.09	12.32	0.10
Standard deviation	3.54	2.98	1.25	1.49	2.01	2.40	0.62

Table 12 Performance of SODNN-Flow on SM2-18–10

Random Seed	Train		Validation		Test		
	RMSE	MAE	RMSE	MAE	RMSE	MAE	R2
RS1	11.03	8.78	7.43	5.65	9.77	6.93	0.80
RS2	3.39	2.47	8.37	6.39	12.42	9.13	0.66
RS3	12.88	10.30	6.84	5.73	16.13	11.85	0.38
RS4	3.72	2.80	9.26	7.86	13.60	10.60	0.48
RS5	9.22	7.50	9.88	8.20	13.40	10.51	0.22
RS6	2.93	2.34	6.85	5.64	19.06	15.57	−1.70
Max – Min	9.95	7.96	3.04	2.56	9.29	8.64	2.50
Average	7.20	5.70	8.11	6.58	14.06	10.77	0.14
Standard deviation	4.00	3.27	1.17	1.06	2.92	2.64	0.84

6.1.6 Testing different values of the SM5 k' parameter via SODNN-Flow

To compare SM5 with other splitting methods in the following sections, various numbers of outer folds (k') are examined while maintaining a fixed inner cross-validation fold of $k = 5$. For each value of k' , hyperparameter tuning is conducted one hundred times per fold. As a result, increasing k' significantly increases the computational cost because each additional outer fold requires one hundred hyperparameter tuning iterations, increasing the resource intensity of the process. On the basis of Table 16, while the best result is observed when $k' = 20$, no pattern can be observed to relate the value of k' to the performance. Doubling the number of folds from $k' = 10$ to $k' = 20$ does not significantly improve the performance, although it requires considerably more computational resources. Therefore, for the remainder of the study, only the case of $k' = 10$ is considered.

6.1.7 Comparing SM5 and SM4 via SODNN-Flow

Now, the additional data splitting method SM5-10–5 is examined via random seeds RS1 through RS6 for comparison with other splitting methods (Table 17). Hyperparameter

tuning is performed in one hundred experiments to choose the proper set of hyperparameters for each scenario.

Comparing Table 17 with other tables that include a test set shows that SM5 consistently delivers a significantly higher average performance and demonstrates nearly zero variance on the test set. This indicates that the random seed has minimal influence on SM5, as its only effect is to determine which data records are included in the test folds simultaneously. Since each data point is included in the test set exactly once, the final reported result reflects the performance of the entire dataset rather than just a subset, ensuring a more robust evaluation and a comparable result for future studies.

Owing to the computational cost of testing all random seeds and splitting methods, SM5-10–5 with random seed 42 (RS6) is selected for the remainder of the study, as it represents the average across the six SM5 RSs (Table 17).

In summary, Fig. 9 illustrates the R^2 performance across six random seed scenarios for different splitting methods that contain at least one test set.

Although the best performances over the experiments are observed for the models with SM2 and SM4 when RS1 is used, both methods exhibit significant fluctuations in other random seeds. In contrast, SM5 maintains consistent results across various splitting methods, achieving a higher average performance. This highlights that while a splitting method

Table 13 Performance of SODNN-CS on SM2-18–10

Random Seed	Train		Validation		Test		
	RMSE	MAE	RMSE	MAE	RMSE	MAE	R2
RS1	0.34	0.28	0.87	0.73	1.22	1.01	0.95
RS2	0.39	0.31	1.01	0.87	0.93	0.84	0.99
RS3	0.55	0.46	0.59	0.45	1.20	0.97	0.96
RS4	0.27	0.21	0.65	0.52	0.69	0.55	0.99
RS5	0.49	0.40	0.61	0.50	1.23	0.97	0.90
RS6	0.36	0.28	0.64	0.53	0.91	0.77	0.99
Max – Min	0.28	0.25	0.42	0.42	0.54	0.46	0.09
Average	0.40	0.32	0.73	0.60	1.03	0.85	0.96
Standard deviation	0.09	0.08	0.16	0.15	0.20	0.16	0.03

Table 14 Performance of SODNN-Flow on SM3-10

Random Seed	Train		Validation		R^2
	RMSE	MAE	RMSE	MAE	
RS1	7.91	6.14	10.07	7.81	0.67
RS2	5.27	3.97	9.46	6.90	0.69
RS3	5.06	3.89	9.07	6.81	0.72
RS4	6.99	5.47	10.37	8.07	0.64
RS5	7.24	5.67	9.58	7.46	0.71
RS6	6.71	5.19	9.72	7.40	0.69
Max – Min	2.85	2.25	1.30	1.26	0.08
Average	6.53	5.06	9.71	7.41	0.69
Standard deviation	1.03	0.84	0.42	0.45	0.03

may achieve high accuracy with a specific random seed, it must demonstrate consistency across different seeds to ensure that the performance is not achieved merely by chance.

6.1.8 Comparing the SODNN-Flow model with H2O models

Given that the R^2 performance of SODNN-Flow on SM5-10–5 is 0.57 on the test set, which may not be considered significant, the H2O models are further assessed under the same conditions (SM5-10–5 and RS6), as presented in Table 18.

Table 15 Performance of SODNN-Flow on SM4-5–10

Random Seed	Train		Validation		Test		
	RMSE	MAE	RMSE	MAE	RMSE	MAE	R2
RS1	8.21	6.51	11.01	8.63	11.44	9.07	0.75
RS2	5.1	4.03	10.99	8.88	13.48	10.87	0.6
RS3	8.24	6.45	10.6	8.39	12.28	9.57	0.64
RS4	6.88	5.74	11.03	9.07	14.32	11.96	0.42
RS5	8.31	6.67	11.4	9.27	12.36	9.91	0.33
RS6	5.47	4.28	8.73	6.78	17.2	13.86	−1.06
Max – Min	3.21	2.64	2.67	2.49	5.76	4.79	1.81
Average	7.04	5.61	10.63	8.5	13.51	10.87	0.28
Standard deviation	1.33	1.07	0.88	0.82	1.89	1.63	0.62

SODNN-Flow, with an R^2 of 0.57, significantly outperforms the H2O library models, with an average performance of 0.37. This finding indicates that the task is challenging and that the SODNN-Flow is more effective in addressing the existing complexity.

6.2 Improving DNN on SM5-10–5

6.2.1 Comparing SODNN and MODNN

The performance of MODNN is now examined owing to the dataset characteristics, particularly the high correlation that exists between the outputs, especially between flow and slump. This approach may improve the performance by allowing the model to learn the relationships between these correlated outputs simultaneously. As mentioned earlier, three scenarios are investigated (Table 19): all targets (MODNN-AT) as outputs, flow and slump (MODNN-FS), and flow and CS (MODNN-FC).

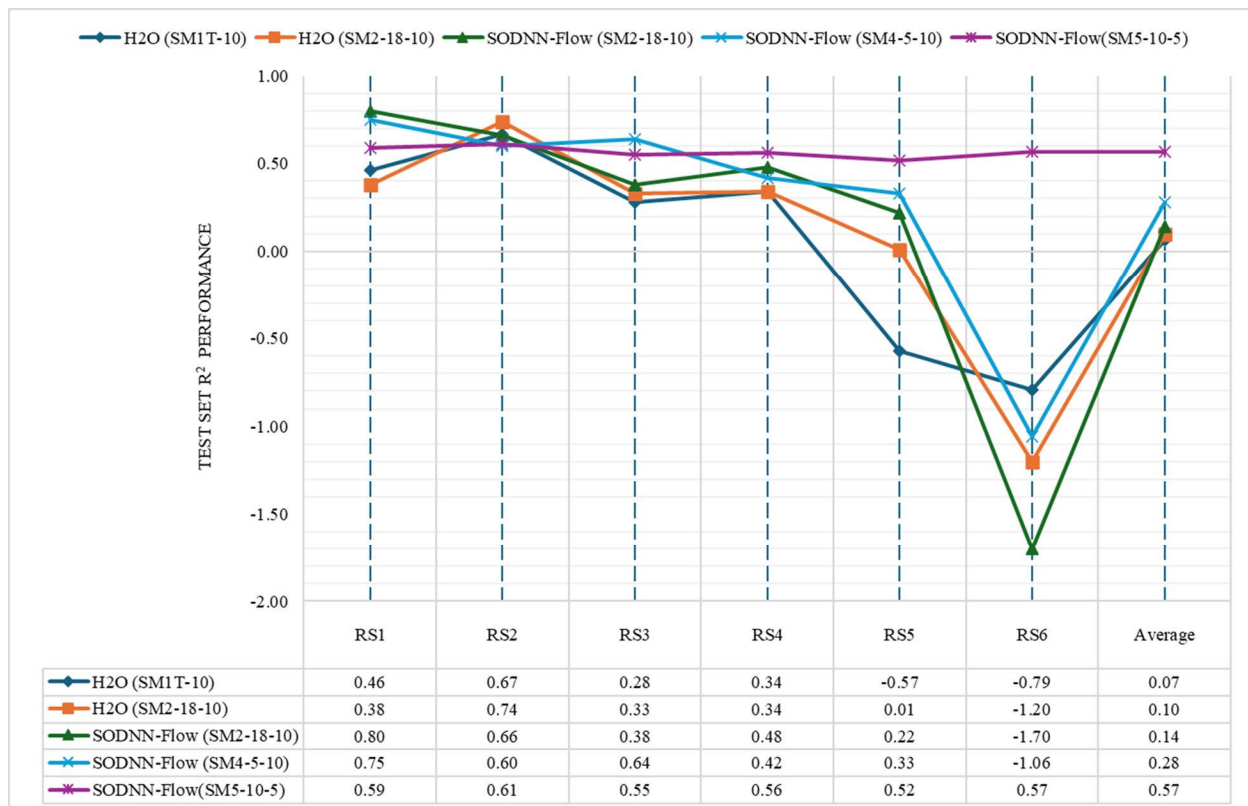
The results in Table 19 indicate that although the average R^2 performance (0.71) of MODNN-AT is higher than those of the other two methods are, the test R^2 performance of flow (0.60) is better when MODNN-FS is used. This is an improvement over the SODNN-Flow performance ($R^2 = 0.57$). This may be due to the high correlation between flow and slump, as they both measure similar concrete properties. This is further

Table 16 Assessing five different values for the k' parameter of SM5 via SODNN-flow

k	Train		Validation		Test		
	RMSE	MAE	RMSE	MAE	RMSE	MAE	R2
5	6.54	5.06	10.27	8.01	12.17	9.47	0.53
10	6.5	5.1	11.82	9.15	10.26	8.02	0.57
15	6.6	5.18	10.48	8.24	11.78	9.26	0.53
20	7.25	5.68	10.56	8.3	10.78	8.7	0.6
25	6.78	5.3	10.9	5.85	11.41	9.5	0.54

Table 17 Performance of SODNN-Flow on SM5-10-5

Random Seed	Train		Validation		Test		
	RMSE	MAE	RMSE	MAE	RMSE	MAE	R2
RS1	6.04	4.68	10.7	8.36	11.48	9.02	0.59
RS2	6.27	4.9	10.7	8.48	11.03	8.6	0.61
RS3	6.83	5.37	10.97	8.64	11.69	9.19	0.55
RS4	6.47	5.02	10.87	8.46	11.82	9.32	0.56
RS5	6.78	5.35	10.9	8.58	12.16	9.77	0.52
RS6	6.5	5.1	11.82	9.15	10.26	8.02	0.57
Max – Min	0.79	0.69	1.12	0.79	1.9	1.75	0.09
Average	6.48	5.07	10.99	8.61	11.41	8.99	0.57
Standard Deviation	0.27	0.24	0.38	0.26	0.62	0.56	0.03

**Fig. 9** Test set R^2 performance for splitting methods that include a test set, evaluated across six random seed scenarios. Lines with less fluctuation indicate lower performance variance

supported by the data in Table 4, where a correlation value of 0.91 is observed between these two variables.

6.2.2 SODNN by using the transfer learning

With an improvement using MODNN, the transfer learning is now evaluated once the slump (SODNN-Slump) is used and once the compressive strength (SODNN-CS) is employed (Table 20).

As previously mentioned, SODNN-Slump and SODNN-CS should first be trained and evaluated on their respective outputs (Slump and CS).

Table 20 shows that SODNN performs significantly better in predicting CS than slump (when the same model and evaluation algorithm are used). This finding suggests that predicting workability is more complex, similar to the comparison of average performance in Table 12 and Table 13, which focused on flow and CS, respectively.

The next step involves transferring the weights and biases of SODNN-Slump and SODNN-CS to predict flow. The corresponding performance results are shown in Table 21.

By examining the results in Table 20 and Table 21, it is observed that although the performance of SODNN-CS is nearly 100%, transferring its weights and biases to predict flow (TLC1-Flow) does not yield a significant improvement. In contrast, while the SODNN-Slump performance is noted as $R^2 = 0.58$, an improvement can be observed in predicting the flow (TLS1-Flow), with $R^2 = 0.62$, compared with all previous results on SM5-10-5. This improvement is possibly due to the high correlation between slump and flow.

Weights and biases are typically imported from an external dataset. However, in this study, the same dataset is used for both tasks. As previously mentioned, in such cases, transfer learning can be implemented in two different ways, as demonstrated by the results in Table 22.

Table 22 shows a significant improvement in TLS2-Flow. This conclusion, however, cannot be readily led to a recommendation for this approach, as it can be considered a form of unfair advantage, particularly due to the high correlation between slump and flow. The reason may be attributed to the fact that when training SODNN-Slump, some data records in the training set may also appear in the TLS2-Flow test set. As a result, the initial weights and biases of TLS2-Flow may have been influenced by the actual slump values during the previous training, leading to an artificially inflated performance.

7 Conclusion

This study conducted a comprehensive comparison of various data splitting methods and examined the effect of changing random seeds on each method to predict the flow properties of concrete via Yeh's dataset as a case study. The aim of

this study was to identify the most suitable splitting method for this predictive task. To achieve this goal, the authors implemented and compared traditional machine learning models and different deep neural network frameworks, including single-output, multi-output, and transfer learning.

The results revealed notable insights into the influence of data splitting methods and random seed selection on the predictive performance of models for concrete flow properties:

- (i) The data-splitting methods that use a single validation set without a separate test set tend to produce overly optimistic performance estimates, which can misrepresent the actual ability of the models to generalize to new data. In contrast, when multiple validation folds are used, such as in cross-validation, the discrepancy between the validation and test set performances is reduced, offering a more reliable indication of the true performance of the model. This underscores the importance of using a distinct test set for final evaluation, particularly in single-validation data-splitting approaches.
- (ii) The choice of random seed significantly impacts the variance in model performance, especially when a portion of the dataset is used as the test set. This occurs because the random seed determines which specific records are included in the test set, influencing the final assessment results. In some instances, changing the random seed may cause the same model to perform either exceptionally well or notably poorly, highlighting how sensitive the test set performance is to the composition of the data selected for evaluation.
- (iii) Compared with the predictions of flow properties and compressive strength when the same dataset and splitting method are used, compressive strength predictions are relatively insensitive to changes in the random seed. In contrast, flow property predictions exhibit high variance, indicating that the impact of random seed selection is task dependent and should be examined accordingly.
- (iv) Most notably, employing the nested cross-validation technique effectively mitigates the random seed effects. Changes in the random seed or the number of outer folds do not significantly affect the model performance. Compared with other splitting strategies, this method yields the best results on the test set, with minimal differences between the validation and test sets. Nested cross-validation offers a robust and comparable approach for different studies by utilizing the entire dataset for evaluation, providing a more realistic assessment of model performance across the whole dataset.

Table 18 Performance of the HO2 library on SM5-10–5

Outer Iteration	Model	Train & Validation		Test		
		RMSE	MAE	RMSE	MAE	R2
1	GBM	11.72	9.67	15.44	13.06	0.43
2	GBM	12.98	10.29	14.37	11.55	0.30
3	GBM	12.80	10.40	13.67	11.01	0.36
4	GBM	12.15	9.70	14.43	11.11	0.14
5	XGBoost	13.33	10.32	9.95	8.73	0.71
6	GBM	11.98	9.15	11.21	9.14	0.62
7	GBM	13.22	10.97	12.50	11.61	0.47
8	Stacked Ensemble	13.12	10.68	11.57	9.95	0.26
9	GBM	12.64	10.35	18.30	14.20	−0.34
10	XGBoost	13.64	11.02	8.33	7.00	0.78
Average		12.76	10.25	12.98	10.74	0.37
Standard deviation		0.60	0.56	2.74	2.00	0.31

Table 19 Performance of multi-output DNNs evaluated on SM5-10–5

Targets	Flow						CS		Slump	Average
	Train		Validation		Test					
	RMSE	MAE	RMSE	MAE	RMSE	MAE	R2			
MODNN-AT	4.63	3.59	10.17	7.77	11.48	8.87	0.58	0.91	0.63	0.71
MODNN-FS	5.63	4.31	10.25	7.81	11.38	8.68	0.6	-	0.59	0.6
MODNN-FC	5.67	4.35	10.45	7.97	11.83	9.09	0.57	0.92	-	0.74

Table 20 Performance of SODNN-Slump and SODNN-CS evaluated by SM5

	Train		Validation		Test		
	RMSE	MAE	RMSE	MAE	RMSE	MAE	R2
SODNN-Slump	3.06	2.33	4.9	3.73	5.74	4.19	0.58
SODNN-CS	0.26	0.21	0.6	0.46	0.7	0.55	0.99

Table 21 Performance of TLS1-Flow and TLC1-Flow on SM5-10–5

	Train		Validation		Test		
	RMSE	MAE	RMSE	MAE	RMSE	MAE	R2
TLS1-Flow	6.59	5.1	10.13	7.85	11.08	8.51	0.62
TLC1-Flow	6.94	5.41	10.62	8.3	11.82	9.38	0.56

Table 22 Performance of two different approaches of transfer learning

	Train		Validation		Test		
	RMSE	MAE	RMSE	MAE	RMSE	MAE	R2
TLS1-Flow	6.59	5.1	10.13	7.85	11.08	8.51	0.62
TLS2-Flow	7.44	5.79	9.61	7.32	9.97	7.64	0.69

Building on the obtained results, comparative analyses based on nested cross-validation with fixed random seeds and consistent configurations of the outer and inner folds further explored the model performance across traditional

machine learning models and diverse deep learning frameworks:

- (i) Comparing the traditional machine learning models with DNN implementations across both nested cross-

validation and simpler data splitting methods, DNN consistently outperforms the non-deep learning models. Although the maximum R^2 score achieved on the test was 0.62, which may not be exceptionally high, it reflects the inherent complexity and characteristics of the dataset rather than the limitations of the modeling approach.

- (ii) In examining the SODNN and MODNN methods, the use of MODNN improves the performance when predicting flow and slump simultaneously (MODNN-FS). This enhancement is likely due to the high correlation between these two properties. However, when compressive strength is included alongside the flow and slump (MODNN-AT), the performance slightly decreases compared with that of MODNN-FS. Nevertheless, the MODNN models do not underperform relative to the SODNN models, even when MODNN-FC is used.
- (iii) The exploration of transfer learning approaches has demonstrated significant potential benefits in improving model performance. Transferring from the SODNN-Slump model to the TLS model results in better performance than transferring from the SODNN-CS model to the TLC model. This outcome is noteworthy because although SODNN-CS achieves nearly 100% accuracy in predicting the compressive strength, SODNN-Slump has a lower performance in predicting the slump, with an R^2 of 0.58. The superior performance of TLS may be attributed to the strong relationship between flow and slump. Ultimately, TLS achieves an R^2 score of 0.62, outperforming the multi-output DNN model, which has an R^2 of 0.60.
- (iv) Two approaches for transfer learning within a single dataset are explored. By allowing records from the training set of the first models (SODNN-Slump or SODNN-CS) to appear in the test set of the second model (as in TL2), an unfair advantage can be introduced over TL1. In TL1, the performance reaches an R^2 score of 0.62, whereas TL2 achieves an R^2 score of 0.69. This observation underscores the necessity of careful data splitting to ensure an unbiased evaluation of model performance.

The findings of this study suggest several ways to improve future research on predicting concrete flow properties:

- (i) **Expansion and creation of larger datasets:** One important direction for future work is to increase the size of datasets by collecting more records related to concrete flow properties. Additionally, adding more

input variables beyond the current 7 features could make predictions more robust and accurate. Given how complex it is to predict flow behavior due to many different factors, larger datasets with more variables can capture more patterns in the data. By gathering more data, researchers can build better models that more accurately reflect real-world situations.

- (ii) **Incorporation of sensitivity analysis and feature importance evaluation:** Future research could explore conducting comprehensive sensitivity analyses and employing explainability methods (e.g., SHAP, LIME) to better understand the relationships between input features and output properties. By assessing the contribution of each feature to model predictions, researchers can identify the most influential variables, enhance model interpretability, and gain deeper insights into the underlying mechanisms driving the model's performance.
- (iii) **Evaluation of alternative machine learning models and techniques via nested cross-validation:** Testing more types of machine learning models and techniques via nested cross-validation could reveal which models perform best. Exploring data augmentation, bootstrapping, and synthetic data generation techniques may help address the challenges of limited training data while mitigating the limitations of small datasets. Another promising avenue for future research is the use of physics-informed neural networks (PINNs) to integrate physical knowledge into future models. PINNs embed governing equations (e.g., constitutive equations for stress and strain in concrete) into the neural network's loss function, ensuring predictions align with physical laws. However, challenges such as computational complexity and the need for precise physical formulations must be addressed. This could improve future model's generalization and interpretability for predicting concrete flow properties. By integrating these approaches with nested cross-validation, future research can take advantage of fair comparisons between different studies, as this method offers a reliable way to test models that incorporate both data enrichment strategies and physical knowledge.
- (iv) **Enhanced hyperparameter tuning strategies:** This study uses a hyperparameter tuning subset size of $t = 100$, which can be expanded. Testing more combinations of hyperparameters could lead to better-performing models.
- (v) **Investigation of additional engineering datasets with limited indices:** Future studies could look at other engineering datasets that, similar to Yeh's dataset, have a small number of records. Studying whether similar patterns and challenges appear in predictions with those

datasets would help extend the findings of this study. This analysis could reveal common issues in modeling engineering properties with limited data and help develop better modeling methods for those cases.

Acknowledgements The authors wish to acknowledge the technical support of the High-Performance Computing Laboratory (HPC Lab), School of Civil Engineering at the University of Tehran.

Author Contribution H.B. contributed to the research by developing the methodology, writing the manuscript, and coding. M.Z. contributed by refining the methodology, validating the code, and reviewing and editing the manuscript. S.M. provided supervision, resources, and methodological input, as well as reviewing and editing the manuscript.

Funding The authors received no specific funding for this study.

Data Availability Sample codes developed in this study are available at: <https://github.com/CHPC-UT/ConcreteML>.

Declarations

Competing Interests The authors declare no competing interests.

References

1. D. Long and B. Magerko. What is AI literacy? Competencies and design considerations. in Proceedings of the 2020 CHI conference on human factors in computing systems. 2020.
2. Mahesh B. Machine learning algorithms-a review. *Int J Sci Res (IJSR)*. 2020;9(1):381–6.
3. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521(7553):436–44.
4. Mishra M, Nayak J, Naik B, Abraham A. Deep learning in electrical utility industry: A comprehensive review of a decade of research. *Eng Appl Artif Intell*. 2020;96:104000.
5. Carvalho TP, Soares FA, Vita R, Francisco RD, Basto JP, Alcalá SG. A systematic literature review of machine learning methods applied to predictive maintenance. *Comput Ind Eng*. 2019;137:106024.
6. Liu B, Vu-Bac N, Zhuang X, Lu W, Fu X, Rabczuk T. AI-DeMat: A web-based expert system platform for computationally expensive models in materials design. *Adv Eng Softw*. 2023;176:103398.
7. Liu B, Lu W. Surrogate models in machine learning for computational stochastic multi-scale modelling in composite materials design. *Int J Hydromechatronics*. 2022;5(4):336–65.
8. Liu B, Vu-Bac N, Rabczuk T. A stochastic multiscale method for the prediction of the thermal conductivity of Polymer nanocomposites through hybrid machine learning algorithms. *Compos Struct*. 2021;273:114269.
9. Liu B, Vu-Bac N, Zhuang X, Fu X, Rabczuk T. Stochastic integrated machine learning based multiscale approach for the prediction of the thermal conductivity in carbon nanotube reinforced polymeric composites. *Compos Sci Technol*. 2022;224:109425.
10. Xia Y, Zhang C, Wang C, Liu H, Sang X, Liu R, et al. Prediction of bending strength of glass fiber reinforced methacrylate-based pipeline UV-CIPP rehabilitation materials based on machine learning. *Tunn Undergr Space Technol*. 2023;140:105319.
11. Liu B, Lu W, Olofsson T, Zhuang X, Rabczuk T. Stochastic interpretable machine learning based multiscale modeling in thermal conductivity of Polymeric graphene-enhanced composites. *Compos Struct*. 2024;327:117601.
12. Liu B, Wang Y, Rabczuk T, Olofsson T, Lu W. Multi-scale modeling in thermal conductivity of polyurethane incorporated with phase change materials using physics-informed neural networks. *Renew Energy*. 2024;220:119565.
13. Tapeh ATG, Naser M. Artificial intelligence, machine learning, and deep learning in structural engineering: a scientometrics review of trends and best practices. *Arch Comput Methods Eng*. 2023;30(1):115–59.
14. Van TN, Quoc TN. Research trends on machine learning in construction management: a scientometric analysis. *J Appl Sci Technol Trends*. 2021;2(02):124–32.
15. Zantalis F, Koulouras G, Karabetsos S, Kandris D. A review of machine learning and IoT in smart transportation. *Future Internet*. 2019;11(4):94.
16. Zhong S, Zhang K, Bagheri M, Burken JG, Gu A, Li B, et al. Machine learning: new ideas and tools in environmental science and engineering. *Environ Sci Technol*. 2021;55(19):12741–54.
17. H.-T. Thai. Machine learning for structural engineering: A state-of-the-art review in Structures. 2022. Elsevier
18. Gamil Y. Machine learning in concrete technology: A review of current researches, trends, and applications. *Front Built Environ*. 2023;9:145591.
19. Li Z, Radlińska A. Artificial intelligence in concrete materials: A scientometric view. In: *Leveraging Artificial Intelligence in Engineering, Management, and Safety of Infrastructure*. CRC Press; 2022. p. 161–83.
20. Li Z, Yoon J, Zhang R, Rajabipour F, Srubar WV III, Dabo I, et al. Machine learning in concrete science: applications, challenges, and best practices. *Npj Comput Mater*. 2022;8(1):127.
21. W. Emad, A.S. Mohammed, R. Kurda, K. Ghafor, L. Cavaleri, S.M. Qaidi, et al. Prediction of concrete materials compressive strength using surrogate models. In *Structures*. 2022. Elsevier.
22. Moein MM, Saradar A, Rahmati K, Mousavinejad SHG, Bristow J, Aramali V, et al. Predictive models for concrete properties using machine learning and deep learning approaches: A review. *J Build Eng*. 2023;63:105444.
23. Laskar AI. Correlating slump, slump flow, vebe and flow tests to rheological parameters of high-performance concrete. *Mater Res*. 2009;12:75–81.
24. Malek J, Kaouther M. Destructive and non-destructive testing of concrete structures. *Jordan J Civ Eng*. 2014;8(4):432–41.
25. M.H. Rafiei, W.H. Khushefati, R. Demirboga, and H. Adeli, 2017 Novel Approach for Concrete Mixture Design Using Neural Dynamics Model and Virtual Lab Concept. *ACI Mater J* 114(1).
26. Garcia C, Onyelowe KC, Valverde Aguirre PE, Ebid AM, Obiano II, Zúñiga Rodríguez MG, et al. Multi-objective optimization of the flow condition of binary constituent net-zero concretes towards carbon neutrality-built environment pathway. *J Build Pathol Rehabil*. 2024;9(1):1–18.
27. Onyelowe KC, Kontoni D-PN, Pilla SRM, Hanandeh S, Ebid AM, Razzaghian Ghadikolaee M, et al. Runtime-based metaheuristic prediction of the compressive strength of net-zero traditional concrete mixed with BFS, FA, SP considering multiple curing regimes. *Asian J Civ Eng*. 2024;25(2):1241–53.
28. Chen N, Zhao S, Gao Z, Wang D, Liu P, Oeser M, et al. Virtual mix design: Prediction of compressive strength of concrete with industrial wastes using deep data augmentation. *Constr Build Mater*. 2022;323:126580.
29. M. Alexander, A. Bentur, and S. Mindess, *Durability of concrete: design and construction*. 2017: CRC Press.

30. Banfill P. Additivity effects in the rheology of fresh concrete containing water-reducing admixtures. *Constr Build Mater.* 2011;25(6):2955–60.
31. E.P. Koehler and D.W. Fowler, Summary of concrete workability test methods. 2003.
32. Neville A, Aitcin P-C. High performance concrete—An overview. *Mater Struct.* 1998;31:111–7.
33. Aydogmus HY, Erdal HI, Karakurt O, Namli E, Turkan YS, Erdal H. A comparative assessment of bagging ensemble models for modeling concrete slump flow. *Comput Concr.* 2015;16(5):741–57.
34. A.C. Rosa, A.W. Hammad, D. Boer, and A. Haddad, 2023 Use of operational research techniques for concrete mix design: A systematic review. *Heliyon* 9(4).
35. P.-C. Aitcin, High performance concrete. 1998: CRC press.
36. Chen L, Kou C-H, Ma S-W. Prediction of slump flow of high-performance concrete via parallel hyper-cubic gene-expression programming. *Eng Appl Artif Intell.* 2014;34:66–74.
37. L. Chen, C. Kou, and S.-W. Ma. Genetic Algorithm Combining Operation Tree (GAOT) for concrete performance. In 5th International Conference on Information Engineering for Mechanics and Materials. 2015. Atlantis Press.
38. Ünlü R. An assessment of machine learning models for slump flow and examining redundant features. *Comput Concr.* 2020;25(6):565–74.
39. A.M. Ivanescu, P. Kranen, and T. Seidl. Hinging hyperplane models for multiple predicted variables. In International Conference on Scientific and Statistical Database Management. 2012. Springer.
40. Hadavandi E, Shahrabi J, Shamshirband S. A novel Boosted-neural network ensemble for modeling multi-target regression problems. *Eng Appl Artif Intell.* 2015;45:204–19.
41. Spyromitros-Xioufis E, Tsoumakas G, Groves W, Vlahavas I. Multi-target regression via input space expansion: treating targets as inputs. *Mach Learn.* 2016;104:55–98.
42. Chen K, Ma Y. Analysis of double single index models. *Scand J Stat.* 2017;44(1):1–20.
43. Lee S, Yang MS, Kang J, Shin SJ. Ensemble variable selection using genetic algorithm. *Commun Stat Appl Meth.* 2022;29(6):629–40.
44. Nabati M, Ghorashi SA, Shahbazian R. JGPR: a computationally efficient multi-target Gaussian process regression algorithm. *Mach Learn.* 2022;111(6):1987–2010.
45. L. Schmid, A. Gerharz, A. Groll, and M. Pauly, 2022 Machine Learning for Multi-Output Regression: When should a holistic multivariate approach be preferred over separate univariate ones? <https://arxiv.org/abs/2201.05340>.
46. Emami S, Martínez-Muñoz G. Deep Learning for Multi-Output Regression using Gradient Boosting. *IEEE Access.* 2024;12:17760–72.
47. Xu Y, Goodacre R. On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *J Anal Test.* 2018;2(3):249–62.
48. Zheng J, Chen Y, Lai Q. PPSFL: Privacy-Preserving Split Federated Learning for heterogeneous data in edge-based Internet of Things. *Futur Gener Comput Syst.* 2024;156:231–41.
49. Bengio, I. Goodfellow, and A. Courville, *Deep learning*. Vol. 1. 2017: MIT press Cambridge, MA, USA
50. Lathuilière S, Mesejo P, Alameda-Pineda X, Horaud R. A comprehensive analysis of deep regression. *IEEE Trans Pattern Anal Mach Intell.* 2019;42(9):2065–81.
51. A. LeNail1, NN-SVG: Publication-Ready Neural Network Architecture Schematics. *Journal of Open Source Software*, 2019.
52. R. Caruana, S. Lawrence, and C. Giles, 2000 Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in neural information processing systems* 13
53. Mehmood F, Ahmad S, Whangbo TK. An Efficient Optimization Technique for Training Deep Neural Networks. *Mathematics.* 2023;11(6):1360.
54. Yang L, Shami A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing.* 2020;415:295–316.
55. Babyak MA. What you see may not be what you get: a brief, non-technical introduction to overfitting in regression-type models. *Psychosom Med.* 2004;66(3):411–21.
56. T. Yu and H. Zhu, 2020 Hyper-parameter optimization: A review of algorithms and applications. <https://arxiv.org/abs/2003.05689>
57. Jesus RJ, Antunes ML, da Costa RA, Dorogovtsev SN, Mendes JF, Aguiar RL. Effect of initial configuration of weights on training and function of artificial neural networks. *Mathematics.* 2021;9(18):2246.
58. S.K. Kumar, 2017 On weight initialization in deep neural networks. <https://arxiv.org/abs/1704.08863>
59. Weiss K, Khoshgoftaar TM, Wang D. A survey of transfer learning. *J Big Data.* 2016;3:1–40.
60. Willmott CJ. Some comments on the evaluation of model performance. *Bull Am Meteor Soc.* 1982;63(11):1309–13.
61. Bates S, Hastie T, Tibshirani R. Cross-validation: what does it estimate and how well does it do it? *J Am Stat Assoc.* 2024;119(546):1434–45.
62. Z. Reitermanova. Data splitting. In *WDS*. 2010. Matfyzpress Prague.
63. P. Refaellizadeh, L. Tang, and H. Liu, 2009 Cross-validation Encyclopedia of database systems, 2009: p. 532–538.
64. Liu B, Penaka SR, Lu W, Feng K, Rebbling A, Olofsson T. Data-driven quantitative analysis of an integrated open digital ecosystems platform for user-centric energy retrofits: A case study in northern Sweden. *Technol Soc.* 2023;75:102347.
65. Schober P, Boer C, Schwarte LA. Correlation coefficients: appropriate use and interpretation. *Anesth Analg.* 2018;126(5):1763–8.
66. T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019.
67. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, et al., 2019 Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32
68. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning. 2015
69. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res.* 2014;15(1):1929–58.
70. Girosi F, Jones M, Poggio T. Regularization theory and neural networks architectures. *Neural Comput.* 1995;7(2):219–69.
71. D.P. Kingma and J. Ba, 2014 Adam: A method for stochastic optimization. <https://arxiv.org/abs/1412.6980> 2014.
72. Sharma S, Sharma S, Athaiya A. Activation functions in neural networks. *Towards Data Sci.* 2017;6(12):310–6.
73. Su X, Yan X, Tsai CL. Linear regression. *Wiley Interdisciplinary Reviews: Computational Statistics.* 2012;4(3):275–94.
74. E. LeDell and S. Poirier. *H2o automl: Scalable automatic machine learning*. in *Proceedings of the AutoML Workshop at ICML*. 2020. ICML San Diego, CA, USA.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.