



Enhancing Concrete Compressive Strength Prediction through Machine Learning Techniques

Hossein Babaei¹, Mohammad Zamani², Soheil Mohammadi³

- 1- B.Sc. in Civil Engineering, High-Performance Computing Laboratory, School of Civil Engineering, College of Engineering, University of Tehran, Iran
- 2- M.Sc. in Civil Engineering, High-Performance Computing Laboratory, School of Civil Engineering, College of Engineering, University of Tehran, Iran
- 3- Professor, High-Performance Computing Laboratory, School of Civil Engineering, College of Engineering, University of Tehran, Iran

smoham@ut.ac.ir

Abstract

Recent advances in artificial intelligence and machine learning are reshaping the field of materials science by enabling predictive frameworks that simplify the design process, substantially reducing reliance on traditional, costly, and labor-intensive testing methods. This study explores the potential of four machine learning techniques: Deep Neural Networks, Random Forest, Light Gradient Boosting Machine, and eXtreme Gradient Boosting, in predicting the compressive strength of concrete. While many studies have applied machine learning to this problem using the same established dataset, a notable gap in the literature is the reliance on validation methods prone to optimistic bias. The primary contribution of this work is the application of a more methodologically rigorous evaluation framework to address this gap. We employ a nested cross-validation (CV) scheme, where the outer loop provides an unbiased measure of model generalization while the inner loop systematically fine-tunes hyperparameters. By mitigating the risk of overfitting, this robust approach provides a comprehensive and reliable benchmark of each algorithm's true performance. These insights offer practitioners trustworthy guidance for forecasting concrete compressive strength, ultimately contributing to more efficient and reliable construction processes.

Keywords: Deep Neural Network, XGBoost, LGBM, RF, Concrete Compressive Strength

1. Introduction

Artificial intelligence (AI) is a field of science that aims to build computational frameworks that can perform tasks that would otherwise require human intelligence to accomplish [1]. A popular subfield of AI is machine learning (ML), which can be broadly defined as a set of computationally intensive methods that use experience to improve performance or make accurate predictions [2]. ML algorithms are specifically designed to find and leverage



patterns in data, with this experience being stored in the form of learned model parameters [3].

Within the field of civil engineering, ML models have been extensively developed to predict physical, mechanical, and durability properties of concrete mixtures [4]. These intelligent models are used for tasks such as predicting the compressive strength of concrete, assessing the workability of fresh concrete, and estimating the carbonation depth for durability analysis [4]. Application of these techniques is driven by the need to solve the complex non-linear relationships between the components of concrete and its resulting properties, which traditional models often struggle to capture [4].

Among the pioneering efforts in applying ML to concrete strength prediction, Yeh [5] made a foundational contribution by demonstrating that Artificial Neural Networks (ANNs) could model the complex, non-linear relationships in concrete compressive strength more accurately than the traditional regression analysis. Crucially, Yeh also provided a comprehensive dataset of 1030 concrete mix instances that became the benchmark for subsequent research in this domain. This dataset has since served as the cornerstone for numerous studies advancing predictive capabilities in concrete strength modeling. For instance, Xu et al. [6] demonstrated that a Random Forest (RF) model achieved superior accuracy compared to both AdaBoost and Support Vector Regression (SVR). Other researchers have focused on optimizing model performance through advanced techniques; Nguyen et al. [7] combined an ANN with Bayesian Optimization (ANN-BO) to systematically tune hyperparameters, while Li et al. [8] employed a Gradient Boosting Regression Tree (GBRT). Echoing these findings, another study by Nguyen et al. [9] used Bayesian optimization to evaluate eight different models, concluding that boosting algorithms like XGBoost and CatBoost were standout performers. Lee et al. [10] built super learner models combining multiple ensemble regressors to predict HPC compressive strength. They outperformed previous methods with lower errors and higher R^2 , offering a reliable tool for mixture design optimization. Naved et al. [11] used hyperparameter-optimized deep neural networks to predict concrete compressive strength. The optimized DNN achieved high accuracy, outperforming regression models and reducing the need for costly lab tests.

A notable gap in the reviewed literature [5–11] is the reliance on standard validation techniques like k-fold cross-validation or simple train-test splits. None of these studies employed nested cross-validation, a more rigorous and robust method for model evaluation. As demonstrated by Babaei et al. [12], nested cross-validation provides a less biased and more reliable estimate for true generalized performance of the model on unseen data.

This study contributes to this growing body of research by conducting a comprehensive comparative analysis of four prominent machine learning algorithms for concrete compressive strength prediction: Random Forest (RF), Light Gradient Boosting Machine (LGBM), XGBoost, and Deep Neural Networks (DNN). To ensure a robust and unbiased performance evaluation, we employ nested cross-validation—a technique notably absent in previous key studies—to provide a more reliable estimate of model generalization capability while mitigating overfitting concerns. Furthermore, systematic hyperparameter tuning is conducted for all models to optimize their predictive performance and ensure a fair comparison. This methodology advances the standard for model validation in this domain by providing a more reliable assessment of true algorithm performance.

The remainder of the paper is organized as follows. First, Sect. 2 provides a brief overview of all models employed in this study, including ensemble methods and deep neural networks (DNNs). Building upon these foundations, Sect. 3 discusses the implementation details and presents the hyperparameter tuning ranges for each model, along with algorithms for their setup and utilization. Subsequently, Sect. 4 describes the characteristics of the dataset used in this case study. Then, Sect. 5 details the findings and provides an analysis of the observed outcomes, which is followed by Sect. 6, which concludes the paper.



2. Methodology

2.1. RF

RF [13] is an ensemble technique that constructs a "forest" of decorrelated decision trees and aggregates their predictions for a more accurate forecast. To ensure diversity, each tree is trained on a bootstrapped sample of data, and at each split, only a random subset of features is considered. The model behavior is governed by tunable hyperparameters that control the number of trees, their structural complexity, and the degree of randomness used in their construction to prevent overfitting.

2.2. LGBM

LGBM [14] is a highly efficient implementation of the gradient boosting framework, where decision trees are built sequentially, with each new tree correcting the errors of its predecessor. Its speed is enhanced by a histogram-based algorithm and a leaf-wise tree growth strategy. To prevent overfitting, the model is managed by a set of hyperparameters that control tree complexity, the pace of the learning process, and the application of stochastic and L1/L2 regularization methods.

2.3. XGBoost

XGBoost [15] is another powerful gradient boosting implementation that sequentially builds an ensemble of decision trees, with each new tree trained to correct the errors of the previous ones. The model complexity is carefully managed by parameters that limit the tree depth and control the conditions for node splitting. It incorporates stochastic elements inspired by bagging and features a sophisticated regularization approach that penalizes both model complexity and large leaf weights, leading to a more robust and generalized model.

2.4. DNN

DNN [16] is a model inspired by the human brain, composed of an input layer, one or more hidden layers, and an output layer. Neurons in each layer process weighted inputs and pass them through an activation function [17], allowing the network to learn complex, non-linear patterns. The network's architecture (e.g., its depth and width) and the training process itself are governed by a set of key hyperparameters. To ensure that the model generalizes well, several regularization techniques are employed to prevent overfitting. No outlier detection or removal was performed on the dataset. For all models, all input features were normalized using the StandardScaler.

3. Implementation

A consistent and rigorous setup is applied for all machine learning models in this study. This involves a nested cross-validation [12] framework configured with 5 outer folds for model evaluation and 5 inner folds for hyperparameter tuning. For each outer fold, an Optuna study, which uses a Tree-structured Parzen Estimator (TPE) sampler [12, 18], is launched to automatically find the optimal hyperparameters by maximizing the coefficient of determination score (R^2) across the inner validation folds.

3.1. Tree Based Algorithms

Hyperparameter optimization for RF, LGBM and XGBoost models is conducted using Optuna. The RF model is implemented with the scikit-learn [19], the LGBM model with the lightgbm library [14], and the XGBoost algorithm with its native library [20]. The respective search spaces for this tuning process are defined in Table 1 for RF, Table 2 for LGBM, and Table 3 for XGBoost.



Table 1. Search space for RF Regressor hyperparameters optimized via Optuna.

Hyperparameter	Description	Search Space
n_estimators	The total number of trees to be grown in the forest.	10 to 300
max_depth	The maximum number of levels a tree can have.	5 to 30
min_samples_split	The minimum number of data points a node must contain to be eligible for splitting.	2 to 8
min_samples_leaf	The minimum number of data points required to form a terminal leaf node.	1 to 5
max_features	The number of features to consider when looking for the best split.	"sqrt", "log2", None

Table 2. Hyperparameter search space for the LGBM model used in the Optuna tuning process.

Hyperparameter	Description	Search Space
max_depth	Sets a limit on the depth of each tree.	3 to 15
min_data_in_leaf	The minimum number of samples that must reside in a terminal node.	5 to 50
learning_rate	Scales the contribution of each new tree.	0.001 to 0.1
feature_fraction	The fraction of features to consider for each tree.	0.5 to 1.0
bagging_fraction	The fraction of data to be used for each tree, enabling subsampling.	0.6 to 1.0
bagging_freq	The frequency (in iterations) to perform bagging.	1 to 7
max_bin	The maximum number of bins that feature values will be bucketed in.	64 to 256
min_gain_to_split	The minimum gain in the loss function required to make a split.	1×10^{-4} to 0.2
lambda_l1	L1 regularization penalty on leaf weights.	1×10^{-4} to 5.0
lambda_l2	L2 regularization penalty on leaf weights.	1×10^{-4} to 5.0

Table 3. The defined search space for the XGBoost model hyperparameters optimized via Optuna.

Hyperparameter	Description	Search Space
n_estimators	The total number of trees to build in the sequence.	200 to 1500
max_depth	The maximum depth a tree can grow to.	3 to 10
learning_rate	The step size shrinkage used to prevent overfitting.	0.0001 to 0.1
subsample	The fraction of training instances to be randomly sampled for each tree.	0.6 to 1.0
colsample_bytree	The fraction of features to be subsampled when constructing each tree.	0.6 to 1.0
gamma	The minimum loss reduction required to make a further partition on a leaf node.	0 to 0.5
min_child_weight	The minimum sum of instance weight (Hessian) needed in a child node.	3 to 20
reg_alpha	L1 regularization term on weights.	1×10^{-4} to 0.5
reg_lambda	L2 regularization term on weights.	1×10^{-4} to 0.5

3.2. Single Output DNN

The Single Output DNN (SODNN) model is implemented in PyTorch [21], following the methodology of Babaei et al. [12]. To prevent overfitting, the training process includes L2 regularization, mini-batch training, dropout, and an early stopping mechanism. Training is capped at a maximum of 2000 epochs and used an early stopping patience of 50 epochs,



halting the process if the training Mean Squared Error fails to decrease by at least 1.0. The model architecture and training parameters are optimized using Optuna, with the specific search space defined in Table 4.

Table 4. Search space for the architectural and training hyperparameters of the SODNN.

Hyperparameter	Search Space
Number of hidden layers	1 to 5
Number of neurons	1 to 256
Learning rate	1×10^{-6} to 1×10^{-2}
Weight decay	1×10^{-6} to 1.0
Dropout rate	0.0, 0.2, 0.5, 0.8
Mini-batch size	0 (full-batch), 64, 128
Optimizer	"Adam", "SGD"[15]

4. Dataset Description

As mentioned in introduction, this study leverages data from key publication by Yeh [5]. The dataset is drawn from Yeh research on modeling the strength of high-performance concrete, containing 1,030 records of various concrete mixtures.

Each column in dataset contains eight quantitative input features that define the concrete composition: Cement, Blast Furnace Slag, Fly Ash, Water, Superplasticizer, Coarse Aggregate, and Fine Aggregate, along with the Age of the sample in days. The target variable in dataset is the experimentally measured Concrete Compressive Strength (CS) expressed in Megapascals (MPa). The detailed statistical properties of the features for the dataset are presented in Table 5.

Table 5. Statistical summary of the Dataset (n=1030).

	mean	std	min	max
Cement	281.17	104.51	102.00	540.00
Blast Furnace Slag	73.90	86.28	0.00	359.40
Fly Ash	54.19	64.00	0.00	200.10
Water	181.57	21.35	121.80	247.00
Superplasticizer	6.20	5.97	0.00	32.20
Coarse Aggregate	972.92	77.75	801.00	1145.00
Fine Aggregate	773.58	80.18	594.00	992.60
Age	45.66	63.17	1.00	365.00
CS	35.82	16.71	2.33	82.60

The resulting correlation matrix [22] for the dataset is presented in Table 6.

Table 6. Pearson correlation matrix for the Dataset (n=1030). The column abbreviations V1-V9 correspond sequentially to the variables in the rows.

	V1	V2	V3	V4	V5	V6	V7	V8	V9
Cement	1.00	-0.28	-0.40	-0.08	0.09	-0.11	-0.22	0.08	0.50
Blast Furnace Slag	-0.28	1.00	-0.32	0.11	0.04	-0.28	-0.28	-0.04	0.13
Fly Ash	-0.40	-0.32	1.00	-0.26	0.38	-0.01	0.08	-0.15	-0.11
Water	-0.08	0.11	-0.26	1.00	-0.66	-0.18	-0.45	0.28	-0.29
Superplasticizer	0.09	0.04	0.38	-0.66	1.00	-0.27	0.22	-0.19	0.37
Coarse Aggregate	-0.11	-0.28	-0.01	-0.18	-0.27	1.00	-0.18	0.00	-0.16
Fine Aggregate	-0.22	-0.28	0.08	-0.45	0.22	-0.18	1.00	-0.16	-0.17
Age	0.08	-0.04	-0.15	0.28	-0.19	0.00	-0.16	1.00	0.33
CS	0.50	0.13	-0.11	-0.29	0.37	-0.16	-0.17	0.33	1.00



5. Result

5.1. Baseline Model Performance on Dataset

To identify the most effective predictive model for this study, a comprehensive evaluation is conducted on four distinct machine learning algorithms: XGBoost, RF, LGBM, and SODNN. For this baseline analysis, all features, including 'age', are treated as numerical inputs. The robustness and generalization capabilities of each model are rigorously assessed through a hyperparameter tuning process with a varying number of hyperparameter combinations (t). The results of this initial comparative analysis are presented in Table 7.

Table 7. Comparative performance of machine learning models with varying numbers of hyperparameter sets (t).

	Train			CV			Test			
Model	t	RMSE	MAE	R ²	RMSE	MAE	R ²	RMSE	MAE	R ²
Xgboost	100	1.460	0.814	0.992	4.234	2.783	0.936	4.005	2.649	0.942
	1000	1.502	0.862	0.992	4.173	2.743	0.938	4.009	2.645	0.942
	10000	1.493	0.854	0.992	4.154	2.752	0.938	3.998	2.651	0.943
RF	100	2.123	1.415	0.984	5.263	3.650	0.901	5.023	3.535	0.909
	1000	2.151	1.452	0.983	5.223	3.664	0.902	5.015	3.540	0.910
	10000	2.162	1.459	0.983	5.221	3.666	0.902	5.016	3.537	0.910
LGBM	100	1.956	1.289	0.986	4.517	3.073	0.927	4.359	2.960	0.932
	1000	1.812	1.175	0.988	4.396	2.987	0.931	4.245	2.883	0.935
	10000	1.910	1.245	0.987	4.356	2.964	0.932	4.301	2.896	0.934
SODNN	100	2.637	1.918	0.975	4.757	3.287	0.919	4.421	3.114	0.930
	1000	2.535	1.824	0.977	4.605	3.186	0.924	4.328	3.032	0.933
	10000	2.394	1.695	0.979	4.544	3.121	0.926	4.294	2.927	0.934

Based on the results from the initial analysis, the XGBoost model demonstrates the most robust performance for the f'_c prediction task. In all test scenarios, XGBoost consistently outperforms the other models, achieving the highest R^2 value and the lowest error metrics, with a peak test R^2 of 0.943.

Conversely, the RF model shows the least predictive accuracy among the tested models. Its performance, with a test R^2 of about 0.910, is lower than the other machine learning methods. The LGBM and SODNN models deliver similar, intermediate results, with test R^2 values in the range of 0.932 to 0.935.

The effect of increasing the number of hyperparameter tuning iterations (t) varies between models, having the most significant effect on the SODNN model based on all three metrics. However, it is noteworthy that for all models, the performance improvement from increasing t from 100 to 10,000 is marginal. This minor gain in predictive accuracy comes at the cost of a substantial increase in computational expense, suggesting that extensive tuning may not be justifiable.

As is characteristic of model fitting, all models yield lower error metrics on the training set than on the cross-validation and test sets. Furthermore, the close alignment in performance between the cross-validation and test sets confirms the effectiveness of the validation strategy in providing a reliable estimate of generalization performance.

5.2. Impact of Treating 'Age' as a Categorical Feature



In this experiment, the 'age' variable is one-hot encoded and treated as a categorical feature. The results are shown in Table 8. Model performance with 'age' as a categorical feature ($t=100$).

Table 8. Model performance with 'age' as a categorical feature ($t=100$).

Model	Train			CV			Test		
	RMSE	MAE	R ²	RMSE	MAE	R ²	RMSE	MAE	R ²
Xgboost	1.478	0.848	0.992	4.248	2.803	0.935	4.042	2.662	0.941
RF	2.180	1.475	0.983	5.235	3.693	0.902	5.063	3.596	0.908
LGBM	2.065	1.369	0.985	4.511	3.089	0.927	4.317	2.916	0.933
SODNN	2.232	1.484	0.982	4.649	3.116	0.922	4.323	2.947	0.933

When 'age' is treated as a categorical feature, the analysis shows a slight decrease in performance for most models compared to when it is treated as a numerical input (at $t=100$). For example, the top-performing XGBoost model experiences a marginal decline in its predictive accuracy. The performance of the RF model also worsens under this condition. The SODNN model, however, is a notable exception, showing a slight improvement in its performance metrics.

Based on these results, and acknowledging the minimal performance shifts, treating 'age' as a continuous numerical variable appears to be the slightly more effective approach for the leading models.

5.3. Impact of Removing the 'Age' Feature

To quantify the predictive importance of the 'age' variable, it is removed from the dataset. The results of this specific study are presented in Table 9.

Table 9. Model performance with the 'age' feature removed ($t=100$).

Model	Train			CV			Test		
	RMSE	MAE	R ²	RMSE	MAE	R ²	RMSE	MAE	R ²
Xgboost	10.348	8.290	0.616	11.917	9.570	0.491	11.716	9.482	0.508
RF	10.472	8.465	0.607	12.121	9.818	0.473	11.930	9.712	0.490
LGBM	10.495	8.433	0.605	11.874	9.557	0.494	11.720	9.490	0.507
SODNN	10.665	8.571	0.592	11.651	9.355	0.513	11.489	9.279	0.527

The removal of the 'age' feature causes a dramatic degradation in performance across all models. The Test R² for the best model, XGBoost, plummets from 0.942 to 0.508. This stark decline underscores the critical importance of the 'age' feature for accurate prediction in this context. However, it is mentionable that in this specific scenario, the SODNN model performs better than the other models, achieving the highest Test R² of 0.527.

Table 10 summarizes the standard deviation of the R² performance over the outer test folds, which serves as a measure of model stability.

Table 10. Performance Stability of Models on Outer Test Folds (R² Standard deviation)

Standard Deviation of R ² Performance on the Outer Test Folds					
	$t=100$	$t=1000$	$t=1000$	'age' as a categorical ($t=100$)	average
Xgboost	0.009	0.010	0.009	0.009	0.009



<i>Rf</i>	0.011	0.011	0.011	0.010	0.011
<i>LGBM</i>	0.009	0.009	0.008	0.009	0.009
<i>SODNN</i>	0.012	0.012	0.013	0.009	0.012

The results indicate a high degree of stability for all models, confirming the robustness of the nested cross-validation approach. As all the standard deviation values are very low, any differences between the models are negligible, indicating that all tested models exhibited a comparably high and robust level of performance stability.

5.4. Feature Importance of Tree-Based Models

To provide deeper insight into the predictive behavior of the ensemble methods, a feature importance analysis was conducted for the tree-based models (RF, LGBM, and XGBoost). Table 11 summarizes the normalized importance scores, which indicate the relative contribution of each input feature to the models' predictions. The feature labels in the table correspond to the concrete mix design parameters as follows: V1 (Cement), V2 (Blast Furnace Slag), V3 (Fly Ash), V4 (Water), V5 (Superplasticizer), V6 (Coarse Aggregate), V7 (Fine Aggregate), and V8 (Age).

Table 11. Feature importance of tree-based models.

		<i>V1</i>	<i>V2</i>	<i>V3</i>	<i>V4</i>	<i>V5</i>	<i>V6</i>	<i>V7</i>	<i>V8</i>
<i>LGBM</i>	t=100	0.26	0.07	0.03	0.13	0.07	0.03	0.06	0.35
	t=1000	0.26	0.07	0.03	0.13	0.07	0.03	0.06	0.34
	t=1000	0.26	0.07	0.03	0.13	0.07	0.03	0.06	0.34
	'age' as a categorical (t=100)	0.27	0.07	0.03	0.13	0.08	0.03	0.06	0.34
<i>RF</i>	t=100	0.31	0.07	0.02	0.11	0.08	0.03	0.04	0.34
	t=1000	0.28	0.07	0.03	0.12	0.08	0.04	0.05	0.34
	t=1000	0.28	0.07	0.03	0.12	0.08	0.04	0.05	0.34
	'age' as a categorical (t=100)	0.25	0.06	0.04	0.13	0.08	0.05	0.06	0.33
<i>Xgboost</i>	t=100	0.17	0.08	0.08	0.11	0.08	0.04	0.05	0.39
	t=1000	0.16	0.08	0.08	0.12	0.08	0.04	0.05	0.38
	t=1000	0.17	0.08	0.08	0.12	0.07	0.04	0.05	0.39
	'age' as a categorical (t=100)	0.17	0.08	0.08	0.12	0.08	0.04	0.05	0.38
<i>average</i>		0.24	0.07	0.05	0.12	0.08	0.04	0.05	0.35
<i>standard deviation</i>		0.05	0.01	0.03	0.01	0.00	0.01	0.01	0.02

The feature importance analysis reveals a clear and consistent hierarchy of predictive factors across all tree-based models. Age (V8) is the most influential feature, with an average importance score of 0.35, followed by Cement (V1) at 0.24 and Water (V4) at 0.12. The remaining features all demonstrated substantially lower predictive influence. This ranking remained remarkably stable across all tested algorithms (LGBM, RF, and XGBoost) and was not significantly affected by extensive hyperparameter tuning or by treating the 'age' feature as a categorical variable. This outcome aligns with fundamental principles of concrete technology, confirming that the models learned physically meaningful relationships where curing time and the primary components of the water-cement paste are the most critical determinants of compressive strength.

6. Conclusion

This study conducted a comprehensive evaluation of several machine learning algorithms, identifying XGBoost as the most effective model for predicting the target variable on the dataset. The XGBoost model consistently demonstrated superior performance, achieving a high degree of accuracy, as evidenced by a peak test R2 of 0.943. Further analysis revealed



that the predictive accuracy was critically dependent on the 'age' feature when treated as a continuous variable, and extensive hyperparameter tuning yielded only marginal returns.

7. Future works

- Expanded Algorithmic Comparison: Broaden the analysis to include other models such as CatBoost, Support Vector Machines (SVM), and hybrid approaches to create a more comprehensive performance benchmark.
- Enhanced Generalizability Testing: Apply the rigorous nested cross-validation framework to other datasets to validate and test the generalizability of the current findings across different types of concrete mixtures.

8. Acknowledgement

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.

9. References

- [1] M. Pankhania, "Artificial intelligence in musculoskeletal radiology: past, present, and future," *Indian Journal of Musculoskeletal Radiology*, vol. 2, no. 89, pp. 89-96, 2020.
- [2] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255-260, 2015.
- [3] D. Bzdok, M. Krzywinski, and N. Altman, "Machine learning: a primer," *Nature methods*, vol. 14, no. 12, p. 1119, 2017.
- [4] Y. Gamil, "Machine learning in concrete technology: A review of current researches, trends, and applications," *Frontiers in Built Environment*, vol. 9, p. 1145591, 2023.
- [5] I.-C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cement and Concrete research*, vol. 28, no. 12, pp. 1797-1808, 1998.
- [6] Y. Xu *et al.*, "Computation of high-performance concrete compressive strength using standalone and ensembled machine learning techniques," *Materials*, vol. 14, no. 22, p. 7034, 2021.
- [7] T.-G. Nguyen, T. D. Tran, and P. D. Nguyen, "Prediction of concrete compressive strength using artificial neural network with bayesian optimization," *Materials and Emerging Technologies for Sustainability*, p. 2550007, 2025.
- [8] D. Li, Z. Tang, Q. Kang, X. Zhang, and Y. Li, "Machine learning-based method for predicting compressive strength of concrete," *Processes*, vol. 11, no. 2, p. 390, 2023.
- [9] K. Le Nguyen, M. Shakouri, and L. S. Ho, "Investigating the effectiveness of hybrid gradient boosting models and optimization algorithms for concrete strength prediction," *Engineering Applications of Artificial Intelligence*, vol. 149, p. 110568, 2025.
- [10] S. Lee, N. H. Nguyen, A. Karamanli, J. Lee, and T. P. Vo, "Super learner machine learning algorithms for compressive strength prediction of high performance concrete," *Structural Concrete*, vol. 24, no. 2, pp. 2208-2228, 2023.
- [11] M. Naved, M. Asim, and T. Ahmad, "Prediction of concrete compressive strength using deep neural networks based on hyperparameter optimization," *Cogent Engineering*, vol. 11, no. 1, p. 2297491, 2024.
- [12] H. Babaei, M. Zamani, and S. Mohammadi, "The impact of data splitting methods on machine learning models: A case study for predicting concrete workability," *Machine Learning for Computational Science and Engineering*, vol. 1, no. 1, p. 21, 2025.
- [13] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5-32, 2001.
- [14] G. Ke *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.
- [15] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785-794.
- [16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436-444, 2015.
- [17] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807-814.



- [18] 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, pp. 2623-2631.
- [19] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research*, vol. 12, pp. 2825-2830, 2011.
- [20] *xgboost: Extreme Gradient Boosting*. (2025). [Online]. Available: <https://github.com/dmlc/xgboost>
- [21] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.
- [22] P. Schober, C. Boer, and L. A. Schwarte, "Correlation coefficients: appropriate use and interpretation," *Anesthesia & analgesia*, vol. 126, no. 5, pp. 1763-1768, 2018.