Predicting the Compressive Strength of Concrete up to 28 Days-Ahead: Comparison of 16 Machine Learning Algorithms on Benchmark Datasets

Farzad Rezazadeh[1], Andreas Kroll[2]

[1,2]Department of Measurement and Control, University of Kassel
Moenchebergstr. 7, 34125 Kassel
E-Mail: {farzad.rezazadeh, andreas.kroll}@mrt.uni-kassel.de

Abstract

Concrete is the most important and widely consumed construction material. Concrete parts are produced by a mixing process, followed by casting and a certain curing time. To assess the quality of concrete, its compressive strength is usually measured (typically after 28 days curing time). Several factors affect the compressive strength of concrete, including environmental factors, the type, quality, and quantity of the constituents, the order of the mixing process, and the curing conditions. Due to the multitude of factors effecting compressive strength and partially known chemical reactions during mixing and curing, in this contribution, data-driven methods are used to model the behavior of the concrete production process. Three different benchmark datasets from the concrete manufacturing field are used for the modeling procedure. 16 typical learning algorithms were selected based on their simplicity and their performance in predicting compressive strength. The results show that 1) repeated cross-validation is more reliable than repeated hold-out in this configuration, 2) the interaction and power terms (2nd order) of the inputs have a positive effect on model prediction, 3) the kernel type of the models is of crucial importance, and 4) gradient boosting and kernel ridge are the most appropriate models for predicting compressive strength.

DOI: 10.58895/ksp/1000151141-4 erschienen in:
Proceedings – 32. Workshop Computational Intelligence: Berlin, 1. - 2. Dezember 2022
DOI: 10.58895/ksp/1000151141 | https://www.ksp.kit.edu/site/books/m/10.58895/ksp/1000151141/

1 Introduction

Conventional concrete (CC) consists of the basic materials Portland cement, fine and coarse aggregate, and water. To achieve higher compressive strength (CS) and better workability of the concrete, the basic materials are supplemented and mixed with additives such as fly ash, silica fume, blast furnace slag, and also superplasticizer in various recipes [1]. Depending on the additives and the type of mixing, high-performance concrete (HPC) or ultra-high performance concrete (UHPC) is produced, as indicated in Table 1. Figure 1 shows an overview of the four steps of the concrete production process. In the first step, the raw materials chosen according to the recipe that are subject to environmental conditions (temperature, humidity), are poured into the mixer according to the specific instructions of the recipe. In the second step, the mixer is first set to a certain speed and duration based on the specific instructions recipe. The result of the second step is fresh concrete. The important factors in fresh concrete are the temperature, electrical conductivity and the amount of air contained. The results of the slump and flow tests can be used as additional describing factors. Lastly, after the curing period (storage of the fresh concrete under certain conditions to make it hard), the final concrete production is prepared. If the compressive strength after curing (lasts typically for 28 days) could be predicted during production (ideally already during the mixing process) off-spec product could be foreseen, correcting action taken and the delivery of off-spec parts be avoided.

1.1 State-of-the-Art Regarding Considered Model Inputs/Features

Rajeshwari et al. [3] attempt to analyze the use of different amounts of fly ash in concrete formulations and conclude that using a large amount of fly ash not only improves the physical properties of concrete, but also reduces greenhouse gas production and is also cost effective. In studying the effects of environmental conditions (during curing), and cement types on the compressive strength of concrete, Farzampour [4] concluded that the temperature and the cement-to-water ratio in extreme weather conditions, are highly critical.

Table 1: CC, HPC, and UHPC can be distinguished by their important characteristics [2].

Concrete	CS	Water/binder	Workability	Cement
Unit	MPa	%	mm	kg/m^3
CC	20-50	0.45-0.65	-	260-380
HPC	50-100	<0.4	455-810	400-700
UHPC	100<	0.2-0.3	260<	800-1000

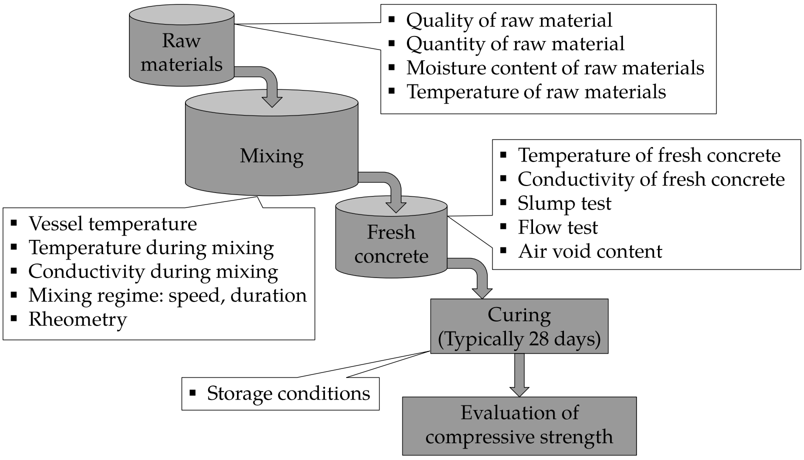
Raw materials
▪ Quality of raw material
▪ Quantity of raw material
▪ Moisture content of raw materials
▪ Temperature of raw materials
Mixing
▪ Temperature of fresh concrete
▪ Conductivity of fresh concrete
▪ Slump test
▪ Flow test
▪ Air void content
▪ Vessel temperature
▪ Temperature during mixing
▪ Conductivity during mixing
▪ Mixing regime: speed, duration
▪ Rheometry
Fresh concrete
Curing (Typically 28 days)
▪ Storage conditions
Evaluation of compressive strength

Figure 1: Process of concrete production and what to consider at each step.

Non-destructive tests such as electrical conductivity [5] and ultrasonic pulse velocity [6] measurement to assess the concrete quality are also used because of their low cost.

1.2 State-of-the-Art Regarding Model Types

Modeling of the concrete production process in order to estimate the concrete quality (usually the compressive strength after 28 days) is generally divided into two categories: traditional modeling (based on empirical relationships) and machine learning. In case of the traditional modeling method, according to Abram's law [7], an empirical relationship based on the ratio of water to cement is used to estimate the compressive strength ($f = b_1/(b_2^{w/c})$, where f is compressive strength after 28 days curing time and b_1, b_2 are empirical constants). In order to complete Abram's law and improve the accuracy of the estimation, Zain et al. [8] proposed a multiple linear regression,

$$f = b_0 + b_1 \frac{w}{c} + b_2 a_1 + b_3 a_2 + c, \tag{1}$$

where f is the compressive strength (after 28 days curing time), w is the water volume, c is the amount of cement, a_1 is the amount of coarse aggregate, a_2 is the amount of fine aggregate, and b_0, b_1, b_2, and b_3 are empirical constants. On the other hand, Zhu et al. [9] have attempted to estimate the compressive strength *f(t)* based only on the curing time t,

$$f(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3. \tag{2}$$

Due to the wide spectrum of effects on compressive strength and the complex and incompletely known chemical reactions during mixing and curing, as well as the inability of classical modeling methods to address a broad range of influencing factors, data-driven methods for modeling the behavior of the concrete production process have prevailed. Ling et al. [10] compared support vector machine (SVM), artificial neural network (ANN), and decision tree (DT) to study the effect of environmental factors on the compressive strength and concluded that the SVM performed better than other methods. On the other hand, Hoang et al. [11] show that the nonparametric and stochastic algorithm Gaussian Process Regression (GP) has a better ability to estimate compressive strength than ANN and SVM. Compared to the above methods, the best estimates for the compressive strength of concrete were obtained using ensemble learning regression [12].

1.3 Scope of Present Work

In this contribution, the performance of models with different structures - from linear to nonlinear, from parametric to nonparametric, and from single model to ensemble - is compared for three concrete benchmark datasets that have different dimensions and a different number of data. At first, the models are trained using two alternative methods: repeated cross-validation and repeated hold-out [13]. The performance variation of the models is assessed based on 20 different initializations for the cross-validation and hold-out methods as well as for each model. The effect of Bayesian optimization is examined on the hyperparameters of the models. The effects of interaction and power terms (2nd and 3rd order) of the inputs are also investigated. For using of the models in online applications, the processing speed of each model is evaluated along with the memory required to estimate the output.

2 Benchmark Datasets Characterization

Concrete production in the laboratory requires purchasing expensive raw materials and handling the manufacturing process. Typically, the production of only one data takes 28 days. This means that the creation of a small dataset with, for example, only 100 data points can take more than half a year dependent on the lab capacity. On the other hand, in the concrete industry, only a few recipes are applied, i.e. there is only little variation in the production process, so that the resulting data may not be very informative. The data available in the literature are usually generated in laboratories and, for simplicity, are based only on changing some constituents, with the environmental variables assumed to be constant [11]. Alternatively, the recipe and the type, quality and quantity of constituents are assumed to be constant and only some environmental factors (usually one or two) are changed [4].

In this paper, three different benchmark datasets from the concrete manufacturing field are used for the modeling procedure. The first dataset (Bdata) [14], was collected from 17 literature sources (laboratory data) and includes 1030 data points. The second dataset (Sdata) [15] with 103 data points and the third dataset (XSdata) [6] with 84 data points originate also from laboratory experiments. Table 2 lists characteristic values for each dataset. As shown, seven factors are identical in Bdata and Sdata. Comparing the minimum and maximum values of the ingredients of Bdata and Sdata, it can be seen that

Table 2: Characterization of the concrete benchmark datasets: Mean, median, standard deviation (STD), minimum (MIN), and maximum (MAX) values of the individual factors.

(a) Bdata (N = 1030)

Ingredient	Unit	Mean	Median	STD	MIN	MAX
Cement	kg/m^3	281.16	272.90	104.50	102	540
Blast furnace slag	kg/m^3	73.89	22	86.27	0	359.40
Fly ash	kg/m^3	54.18	0	63.99	0	200.10
Water	kg/m^3	181.56	185	21.35	121.80	247
Superplasticizer	kg/m^3	6.20	6.40	5.9	0	32.20
Coarse aggregate	kg/m^3	972.91	968	77.75	801	1145
Fine aggregate	kg/m^3	773.58	779.50	80.17	594	992.60
Age	day	45.66	28	63.16	1	365
Compressive strength	MPa	35.81	34.44	16.70	2.33	82.6

(b) Sdata (N = 103): The slump and flow tests assess the fluidity (looseness or stiffness) and the consistency of the fresh concrete, respectively.

Ingredient	Unit	Mean	Median	STD	MIN	MAX
Cement	kg/m^3	229.89	248	78.87	137	374
Blast furnace slag	kg/m^3	77.97	100	60.46	0	193
Fly ash	kg/m^3	149.01	164	85.41	0	260
Water	kg/m^3	197.16	196	20.20	160	240
Superplasticizer	kg/m^3	8.53	8	2.80	4.40	19
Coarse aggregate	kg/m^3	883.97	879	88.39	708	1050
Fine aggregate	kg/m^3	739.60	742.70	63.34	640.60	902
Slump test	cm	18.04	21.50	8.75	0	29
Flow test	cm	49.61	54	17.56	20	78
Compressive strength	MPa	36.03	35.52	7.83	17.19	58.53

(c) XSdata (N = 84)

Ingredient	Unit	Mean	Median	STD	MIN	MAX
Blast furnace slag/binder	%	0.30	0.30	0.22	0	0.60
Ultrasonic pulse velocity	km/s	4.14	4.18	0.39	3.16	4.81
Age	day	73.42	56	65.01	3	180
Compressive strength	MPa	30.45	30.50	12.94	6.32	54.14

Bdata generally uses a larger range to create each sample. In general, Bdata has a high dimenson with large data points, Sdata has almost the same high dimenson as Bdata but with significantly fewer data points, and XSdata has low dimensions and also less data points.

3 Used Data-Driven Modeling Methods

To model concrete manufacturing process, linear vs. nonlinear single models and nonlinear ensemble models are investigated. In the ensemble structure, instead of using a single model to predict the system behavior, a combined structure with multiple algorithms is used to explore the data from different angles and achieve a better understanding of the patterns by combining their results into a final prediction [12]. In this contribution, only two ensemble structures (bagging and boosting) are considered with decision trees as the base learners. All used models originate from the Sklearn [16], Xgboost [17], and LightGBM [18] Python's libraries. A PC with Intel(R) Core(TM) i9-10900X CPU and 64.0 GB RAM is used. Tables 4 and 5 give an overview of the used models with the considered hyperparameters (HPs).

Bayesian optimization, from the Skopt [16, 32] library in Python, is used to determine the optimal hyperparameters. Bayesian optimization is a non-derivative and fast optimization technique that searches to find the global minimum. Bayesian optimization uses an optimization procedure to create a probability model, also known as a surrogate model of the objective function, and selects a hyperparameter in each iteration based on the value of an acquisition function in the previous iteration for the probability function [33]. The best hyperparameter is selected based on the best value of the acquisition function during the whole procedure.

To achieve a more realistic and also generalized performance measure for each algorithm, 20 times repeated hold-out and repeated cross-validation [13] are applied to split each of the three datasets into training and test datasets. The same dataset splits are applied for each model training. In each repetition, a random initialization is used. Finally, the average performance of each algorithm in the 20 runs is calculated (Algorithm 1 for the repeated hold-out).

Table 4: Overview of linear and nonlinear single models/algorithms

	Model	Ab.	Description	HP	Re.
Linear models	Linear Regression	LI	Model with tunable coefficients minimizing the difference between the estimated and true outputs	None	[12]
	Lasso	LS	Model that provides a sparse remedy based on the L1 penalty	$\alpha1$ (L1 coefficient)	[19]
	Ridge	RG	Model that provides a sparse remedy based on the L2 penalty	$\alpha2$ (L2 coefficient)	[20]
	Kernel Ridge	KR	Combination of RG and kernel trick	$\alpha2$, kernel	[21]
	Elastic Net	EN	Model that provides a sparse remedy based on the L1 and L2 penalties	$\alpha1$, $\alpha2$	[22]
Nonlinear single models	Support Vector Machine	SV	Finding the line/hyperplane based on minimizing the distance between predicted and true values within highest confidence margin	$\alpha2$, kernel, polynomial kernel degree	[23]
	K-nearest neighbor	KN	Output based on the average of the k-values of the nearest neighbor points	Number and distance type of neighbors	[24]
	Gaussian Process Regression	GP	Non-parametric Bayesian modeling	Kernel	[25]
	Decision Tree	DT	Non-parametric model for estimating output based on straightforward decision rules	Depth of tree	[26]

4 Results

It should be noted that after investigating the correlation between the factors of each dataset, it is found that only in Sdata there is a strong correlation just between slump and flow (90 %). So slump is chosen as an input for modeling, since the model performances are almost the same when training the models with and without flow.

Table 5: Overview of nonlinear ensemble models/algorithms

	Model	Ab.	Description	HP	Re.
Nonlinear ensemble models	Random Forest	RF	Model based on the different DTs using bootstrap replicas	Number of trees	[27]
	Extra Tree	ET	Model based on the different DTs using whole dataset	Number of trees	[28]
	AdaBoost	AB	Prediction based on decision stumps grounded on DT with only one node and two leaves	Number of trees, learning rate	[29]
	Gradient Boosting	GB	Based on DT and gradient descent optimization, starting from a single leaf as opposed to AB (AB starts from a stump)	Number of trees, learning rate	[30]
	Stochastic Gradient Boosting	XB	Based on GB and regularization principle to avoid overfitting. It increases the leaves in the horizontal direction	Number of trees, depth of tree, $\alpha1$, $\alpha2$	[17]
	Light Gradient Boosting	LB	LB increases the leaves in the vertical direction, which leads to a better and also faster performance than XB	Number of trees, learning rate	[18]
	Histogram Gradient Boosting	HB	Based on GB with a preprocessing technique through discretization to group data	Depth of tree, learning rate, $\alpha2$	[31]

The results of the 20 repetitions of hold-out and cross-validation are shown in Figures 2, 3, and 4 for Bdata, Sdata, and XSdata, respectively. The abbreviations of the model types are defined in Tables 4 and 5. The evaluation criterion used here is R^2 because it is simple, standardized, and the most widely used criterion in predicting compressive strength:

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2} \ , \tag{3}$$

where $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$ and $\hat{y}$ is predicted y. With the hold-out method, the performance of the model depends on the choice of the data for training and testing. This dependence is evident in the modeling results of all three datasets. For 20

Algorithm 1.: Repeated hold-out procedure

```
model ← one choice from 16 available algorithms
dataset ← one choice from Bdata, Sdata, or XSdata
results ← []
splitSize ← 10 % for test data (90 % for training data)

  for i ← 0:20 (20 times hold−out)
      seed ← initialize with random values
      Train, Test ← Splitting dataset based on seed
      optimHypers ← []
      for Fold ← 10 (Bayesian optimization loop)
          train, test ← Splitting Train based on seed
          optimHyper ← optimal hyperparameters (model)
          optimHypers ← [optimHypers, optimHyper]
      end
      OHP ← best hyperparameters from optimHypers
      trainedModel ← fit model with Train (seed, OHP)
      result ← evaluate trainedModel with Test
      results ← [results, result]
  end
  RESULT ← mean of results
  return RESULT
```

repetitions of the hold-out procedure, the performance varies in a large range. For example, LI performance varies from 45 % to 66 % for Bdata, from 75 % to 94 % for Sdata, and from 60 % to 93 % for XSdata. In reviewing the literature estimating the concrete behavior, in most of the papers the results are obtained using the (one-time) hold-out method, without any other initial randomization of the model training. Such results are not reliable and reproducible, since the best performance of the model could be selected by chance. For example, in this contribution, SV performance for the repeated hold-out (20 runs) is equal to $R^2_{max} = 73.94$ %, $R^2_{mean} = 63.76$ %, and $R^2_{min} = 53.16$ % for Bdata and $R^2_{max} = 94.28$ %, $R^2_{mean} = 86.08$ %, and $R^2_{min} = 71.49$ % for Sdata. However, in [34] the reported SV performance for Bdata is $R^2 = 73.98$ %, and in [35] for Sdata is $R^2 = 85.50$ % (only one hold-out). In such a case, the performance of the model with real data may show a large deviation. To reduce the impact of this problem, the repeated hold-out method is used. However, the disadvantage of the repeated hold-out method is the possibility that some data never appear in the training or in the testing process. Cross-validation is an alternative, but to obtain results with higher confidence, repeated cross-validation is recommended [13]. In Figures 2, 3, and 4, the performance of each model based on repeated cross-validation varies in a smaller range and also generally has a smaller median than that of the hold-out method. This is because in each boxplot, each point of 20 runs of cross-validation represents an average of 10 training-test procedures (10-fold corss-validation), but in contrast, in each boxplot of the hold-out method, each point of 20 runs represents only one training-test procedure.

As shown in Figures 2, 3, and 4, the performance of the linear models is quite weak for Bdata, where the dimension of the input space and the number of data points are high, and also in XSdata, where both the dimension and the number of data are low. But these linear models have shown good performance for Sdata, which has a high dimension like Bdata and a small number of data like XSdata. As a result, KR achieved the best performance for Sdata.

The ensemble models used are all developed based on decision trees, and in this context, the ensemble models perform better on average than their base learner. An excellent performance of ensemble models occurs for Bdata, but these models have not shown good results for Sdata, especially when compared to linear models. The reason is that these models require a large amount of data for training. This data requirement also depends on the dimension of the input space, so ensemble models perform much better in XSdata, which has almost the same amount of data but a much smaller dimension of the input space than Sdata. In general, the biggest challenge is the high dimension and small number of data in Sdata compared to the other two datasets. In such a situation, ensemble models based on decision trees and other conventional nonlinear models are not a good choice. On the other hand, GP adapts well to the small amount of data. By applying the kernel trick to the linear model (KR), i.e., by better simulating the nonlinear hidden pattern of the data in the model, better results can be obtained for Sdata than with other models.

To examine the effect of Bayesian optimization, the entire process of hold-out and cross-validation (with 20 runs) is performed both with and without the hyperparameter optimization loop. The average of the 20 runs of each model

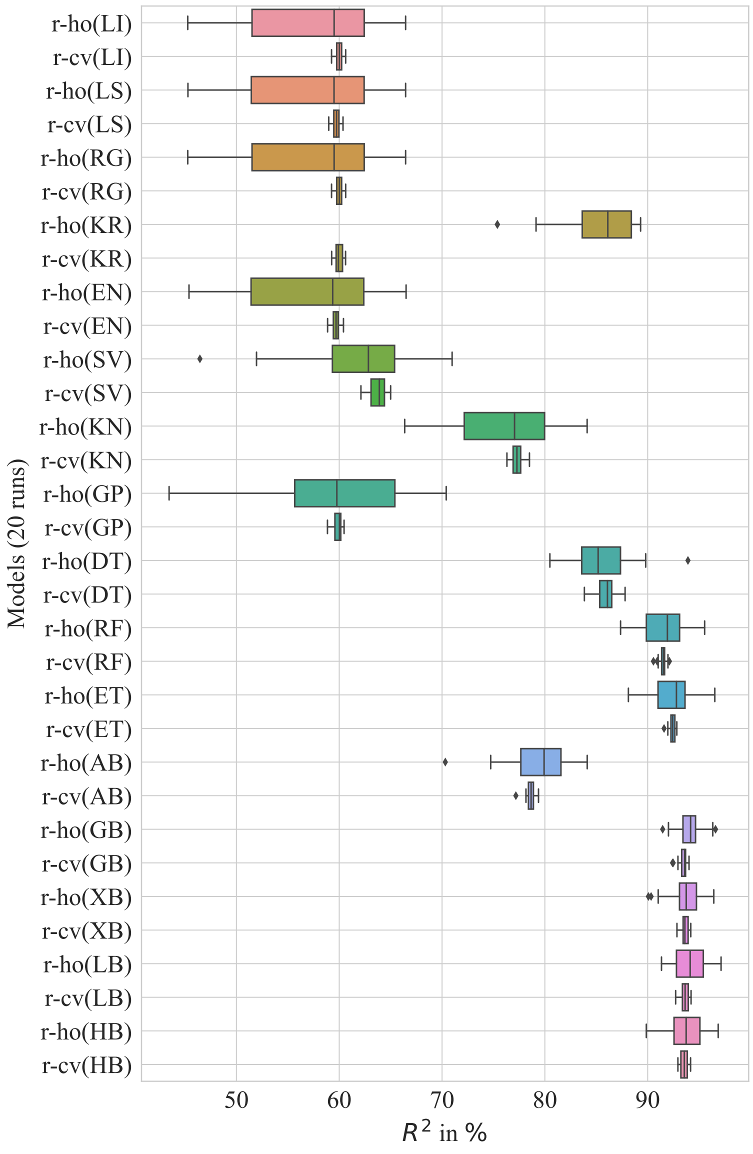

Figure 2: Repeated hold-out (r-ho(model)) vs. repeated cross-validation (r-cv(model)), Bdata

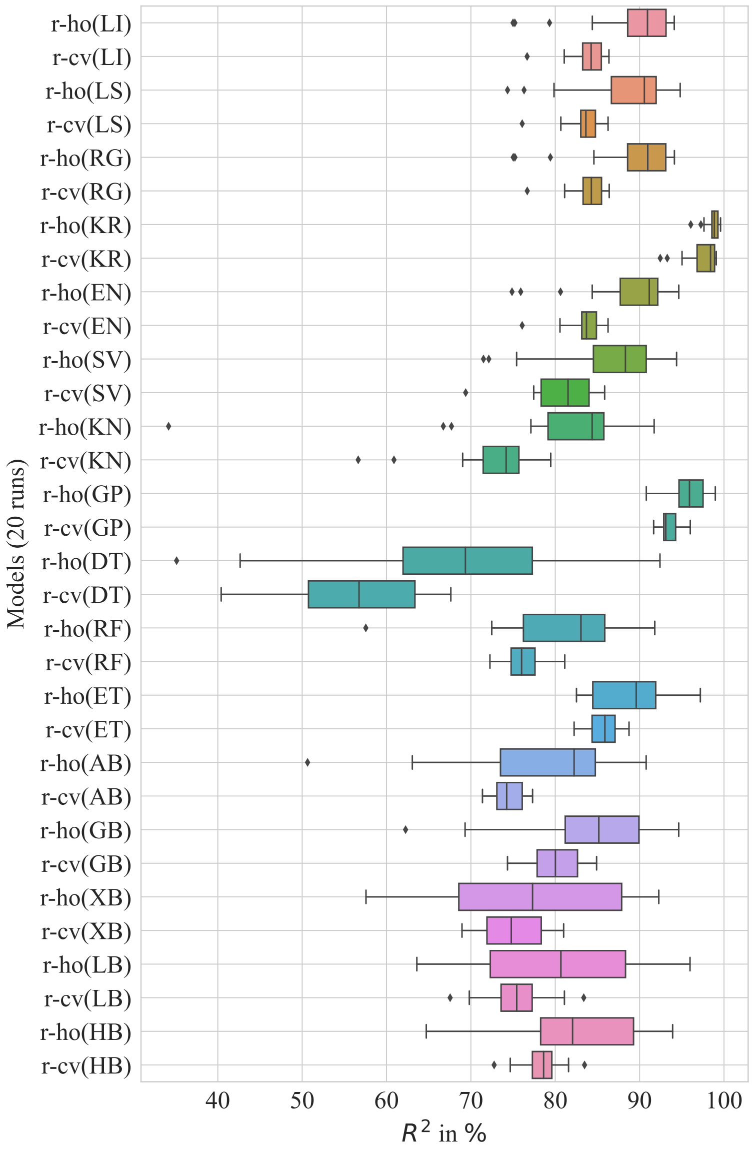

Figure 3: Repeated hold-out (r-ho(model)) vs. repeated cross-validation (r-cv(model)), Sdata

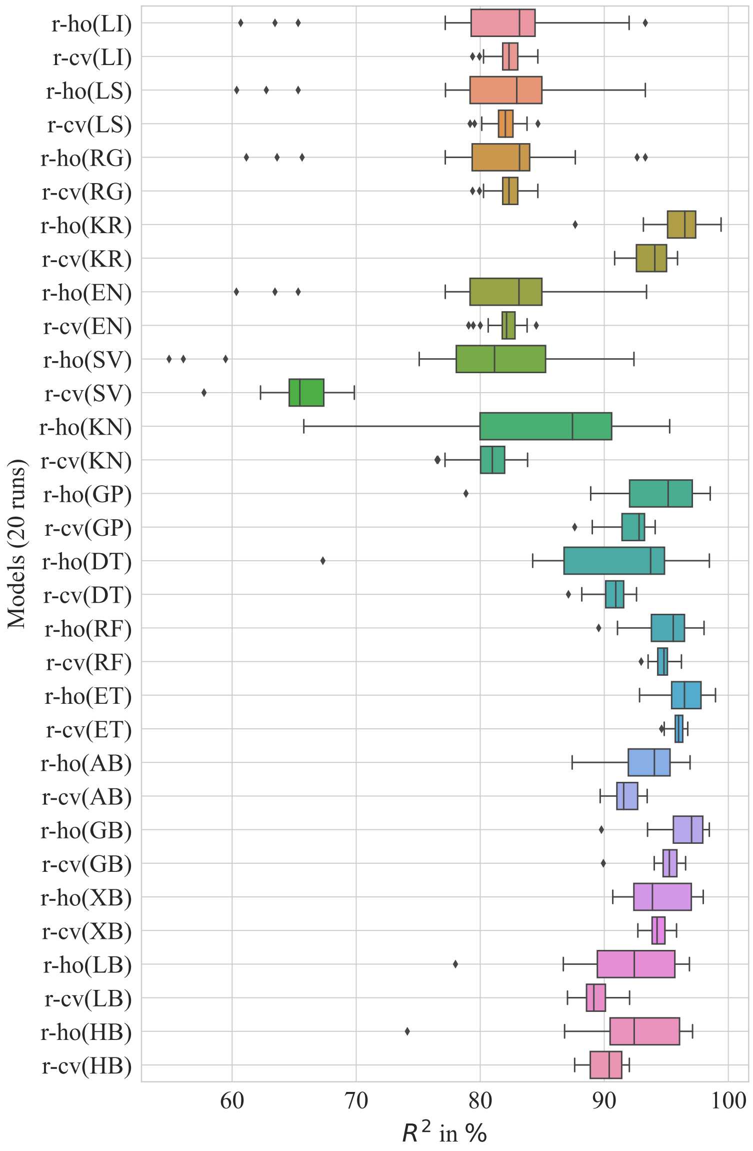

Figure 4: Repeated hold-out (r-ho(model)) vs. repeated cross-validation (r-cv(model)), XSdata

in all three datasets is presented in Table 6. In general, there is not much difference between the time required to train a model without the optimization loop in the three datasets. For example, KN takes 0.5, 0.4, and 0.4 seconds as training time for Bdata, Sdata, and XSdata; AB takes 0.2, 0.2, and 0.1; and HB needs 0.5, 0.4, and 0.4 seconds, respectively. But there is a big difference in the training time needed, if KN is utilized in the optimization loop. For example, KN needs 49 seconds for Bdata, 41 seconds for Sdata and 42 seconds for XSdata. AB takes 80, 61, 60 and HB takes 151, 122, and 113 seconds, respectively. The Bayesian optimization loop can increase the training time by more than 100 times. Now the question arises whether such an optimization loop is necessary in the modeling process. From Table 6, it can be concluded that Bayesian optimization improves the models where the kernel types and the number of neighbors are considered in the optimization loop (see the performance of KR, SV, GP, and KN). On the other hand, the linear models (RG and EN) where the regularization coefficient is a hyperparameter, or even the models based on the decision trees, do not differ much when the optimization is used.

The next step is to investigate the effect of the polynomial forms (interaction and power terms with degree two and three) of the input space on the performance of the models. Transferring the data into quadratic polynomial terms (into the interaction and also the power terms) resulted in a significant improvement in the average accuracy of the linear models. For example, the average LI performance in Bdata increased from 57.34 % to 68.31 % and in Sdata from 88.98 % to 97.23 %. In contrast, for the nonlinear models, this transfer of input space had a slightly positive effect on Bdata (KNN from 75.91 % to 76.80 %) and even a negative effect on Sdata (AB from 78.37 % to 67.59 %). These engineered nonlinear patterns (polynomial terms of 2nd degree) help linear models to recognize the nonlinear behavior of the data. On the other hand, nonlinear models do not require such feature engineering, and using higher order input terms introduces more complexity in the search space for models and can weaken the performance. Transferring the inputs to polynomial space with degree 3 had a negative effect on the performance of the models in all three datasets.

Table 6: Mean values of 20 runs of repeated hold-out (r-ho(model)) and repeated cross-validation (r-cv(model)), with (R^2 in %) and without (R_0^2 in %) Bayesian optimization.

	Bdata		Sdata		XSdata	
Model	R^2	R_0^2	R^2	R_0^2	R^2	R_0^2
r-ho(LI)	57.34	-	88.98	-	80.76	-
r-cv(LI)	**59.97**	-	**83.92**	-	**82.19**	-
r-ho(LS)	57.32	57.47	88.39	88.80	80.73	77.65
r-cv(LS)	**59.71**	59.82	**83.39**	82.64	**81.95**	73.56
r-ho(RG)	57.34	57.52	89	88.63	80.85	83.94
r-cv(RG)	**59.97**	59.84	**83.96**	82.91	**82.19**	81.05
r-ho(KR)	85.03	57.54	98.71	88.41	96.03	59.94
r-cv(KR)	**59.94**	59.90	**97.49**	82.78	**93.77**	51.80
r-ho(EN)	57.36	57.50	88.73	88.36	80.83	42.54
r-cv(EN)	**59.65**	59.84	**83.50**	82.95	**82.04**	36.29
r-ho(SV)	61.82	24.19	86.08	2.57	78.87	33.19
r-cv(SV)	**63.76**	25.01	**80.98**	2.03	**65.65**	23.63
r-ho(KN)	75.91	68.97	79.98	74.46	84.75	81.88
r-cv(KN)	**77.31**	71.19	**72.75**	59.49	**80.58**	77.81
r-ho(GP)	58.95	3.26	95.69	2.18	93.95	88.63
r-cv(GP)	**59.87**	3.14	**93.45**	3.12	**92.13**	87.97
r-ho(DT)	85.75	85.44	68.33	57.33	90.56	92.96
r-cv(DT)	**85.91**	85.76	**55.93**	56.29	**90.63**	89.54
r-ho(RF)	91.60	91.73	81.13	79.05	95	96.28
r-cv(RF)	**91.48**	91.62	**76.15**	75.54	**94.70**	94.42
r-ho(ET)	92.55	92.45	88.89	87.82	96.43	97.02
r-cv(ET)	**92.42**	92.53	**85.72**	85.22	**95.93**	95.82
r-ho(AB)	79.39	78.43	78.37	78.55	93.27	94.94
r-cv(AB)	**78.60**	78.58	**74.43**	72.71	**91.69**	92.30
r-ho(GB)	94.10	90.06	84.56	83.62	96.32	96.87
r-cv(GB)	**93.45**	90.49	**79.87**	80.52	**94.99**	95.07
r-ho(XB)	93.68	93.50	77.44	78.55	94.37	95.36
r-cv(XB)	**93.63**	93.46	**75.23**	74.57	**94.34**	94.18
r-ho(LB)	94	93.41	80.31	83.34	91.71	91.96
r-cv(LB)	**93.64**	93.31	**75.60**	76.24	**89.23**	89.59
r-ho(HB)	93.74	93.40	82.44	82.98	91.93	92.42
r-cv(HB)	**93.53**	93.36	**78.27**	77.18	**90.14**	90.51

It should also be kept in mind that transferring the inputs to the higher order polynomial space significantly increases the time required in the optimization loop, in the training process, and in the testing phase, so a trade-off between time and accuracy should be considered when applying such techniques.

Table 7 illustrates three important attributes for the evaluation of each model under online application, i.e., performance (R^2), prediction time (P_t), and memory requirement for prediction (P_m). For Bdata, GB model with $R^2 = 93.45$ %, $P_t = 1.40$ ms, and $P_m = 18114$ KiB performs better than the other models in all three factors. GB is followed by the LB models with high performance ($R^2 = 93.64$ %) and acceptable memory requirement ($P_m = 24082$ KiB) but high prediction time ($P_t = 16.71$ ms), and AB ($R^2 = 78.60$ %, $P_t = 3.59$ ms, $P_m = 216247$ KiB), and KN ($R^2 = 77.31$ %, $P_t = 2.50$ ms, $P_m = 67315$ KiB) with acceptable prediction time and memory requirement but lower performance. In Sdata, KR with $R^2 = 97.49$ %, $P_t = 17.65$ ms and $P_m = 10351$ KiB are able to be the best choice, but on the other hand, GB with $R^2 = 79.87$ %, $P_t = 0.62$ ms, and $P_m = 3440$ KiB needs less speed and memory than KR, but its performance is weaker. Finally, for XSdata, GB with $R^2 = 94.99$ %, $P_t = 0.62$ ms, and $P_m = 3044$ KiB is the best choice for online application.

5 Conclusions

In this study, 16 data-driven modeling algorithms of linear and nonlinear, parametric and nonparametric type, and ensembles are investigated for predicting compressive strength of concrete. For this, three datasets with different number of dimensions and different number of data are used. Based on the results of the repeated hold-out and repeated cross-validation methods, it is recommended to use the repeated cross-validation in the training process when the required high computational power is available. The results vary less, which means that it provides a more reliable assessment of the model performance. It is also recommended to use Bayesian optimization only for models with kernels (KR, SV, and GP) and for KN. In cases where the regularization coefficient or the number and depth of the trees are hyperparameters for the model, Bayesian

Table 7: Performance of the models for the online application based on R^2 (in %) vs. prediction time (P_t in ms) vs. memory consumed for prediction (P_m in KiB).

	Bdata			Sdata			XSdata		
Model	R^2	P_t	P_m	R^2	P_t	P_m	R^2	P_t	P_m
LI	59.97	16.56	14969	83.92	22.81	4200	82.19	17.18	4596
LS	59.71	18.90	14958	83.39	23.90	3984	81.95	23.43	4036
RG	59.97	6.09	14895	83.96	21.09	3488	82.19	22.34	3092
KR	59.94	29.53	772124	97.49	17.65	10351	93.77	19.53	8172
EN	59.65	10	14963	83.50	24.37	3552	82.04	19.21	3156
SV	77.31	2.50	67315	80.98	436.09	71723	65.65	435.31	70731
KN	59.87	16.25	996430	72.75	5.46	34320	80.58	5.62	24136
GP	91.48	69.53	106916	93.45	50.78	99583	92.13	45.15	101366
DT	63.76	15.46	15089	55.93	21.71	4648	90.63	18.28	4252
RF	85.91	25.46	18102	76.15	13.75	3432	94.70	17.34	3036
ET	92.42	28.59	94573	85.72	56.56	84839	95.93	55.78	87991
AB	78.60	3.59	216247	74.43	2.65	24079	91.69	2.34	22671
GB	93.45	1.40	18114	79.87	0.62	3440	94.99	0.62	3044
XB	93.63	51.71	17183	75.23	55	17711	94.34	50.93	17711
LB	93.64	16.71	24082	75.60	32.50	12010	89.23	36.87	10578
HB	93.53	36.25	57485	78.27	58.28	49041	90.14	54.37	49585

optimization is not required and the model can be used directly as a tool with default values for the hyperparameters. The possible explanation are that the default values of the hyperparameters of the models used are generally optimal or that the types of the surrogates (Gaussian process), the acquisition functions (Expected improvement) and also the number of iterations (50) used for Bayesian optimization are not the best choices and other selections should be used for such cases. The consideration of the quadratic polynomial terms is recommended, especially in case of small datasets and for linear models (it is not recommended for nonlinear models). This concept of the polynomial terms can be developed specifically for the ensemble methods based on other base learners (not only DTs). Finally, due to the high performance and speed of GB for Bdata and XSdata, and the high performance and acceptable speed of KR for Sdata (if only limited resources are available, GB is better choice than KR for Sdata), these models are recommended as superior models for the considered application.

Acknowledgment

This project (HA project no. 1011/21-13) is financed with funds of LOEWE – Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz, Förderlinie 3: KMU-Verbundvorhaben (State Offensive for the Development of Scientific and Economic Excellence).

References

[1] D. Dutta and S. V. Barai. “Prediction of compressive strength of concrete: machine learning approaches”. In: *In Recent Advances in Structural Engineering* 1. pp. 503-513. 2019.

[2] M. T. Marvila, A. R. G. de Azevedo, P. R. de Matos, S. N. Monteiro and C. M. F. Vieira. “Materials for production of high and ultra-high performance concrete: review and perspective of possible novel materials”. In: *Materials* 15. pp. 4304. 2021.

[3] R. Rajeshwari and S. Mandal. “Prediction of compressive strength of high-volume fly ash concrete using artificial neural network”. In: *Sustainable Construction and Building Materials* 20.4. pp. 471-483. 2019.

[4] A. Farzampour. “Compressive behavior of concrete under environmental effects”. In: *Sustainable Construction and Building Materials* pp. 92-104. 2019.

[5] A. Akhnoukh. “Overview of Concrete Durability Evaluation Using Electrical Resistivity”. In: *Collaboration and Integration in Construction, Engineering, Management and Technology* pp. S. 9-14. 2004.

[6] S. Czarnecki, M. Shariq, M. Nikoo and Ł. Sadowski. “An intelligent model for the prediction of the compressive strength of cementitious composites with ground granulated blast furnace slag based on ultrasonic pulse velocity measurements”. In: *Measurement* 172 pp. 108951. 2021.

[7] S. Propovics. “Analysis of concrete strength versus water-cement ratio relationship”. In: *Materials Journal* 87.5. pp. 517-529. 1990.

[8] M. F. M. Zain, S. M. Abd, K. Sopian, M. Jamil and A. I. Che-Ani. “Mathematical regression model for the prediction of concrete strength”. In: *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering* 10. 2008.

[9] W. C. Zhu, J. G. Teng and C. A. Tang. “Mesomechanical model for concrete. Part I: model development”. In: *Magazine of concrete research* 56.6, pp. 313-330. 2004.

[10] H. Ling, C. Qian, W. Kang, C. Liang and H. Chen. “Combination of Support Vector Machine and K-Fold cross validation to predict compressive strength of concrete in marine environment”. In: *Construction and Building Materials* 206. pp. 355-363. 2019.

[11] N. D. Hoang, A. D. Pham, Q. L. Nguyen and Q. N. Pham “Estimating compressive strength of high performance concrete with Gaussian process regression model”. In: *Advances in Civil Engineering*. 2861380. 2016.

[12] J. Yu, R. Pan, and Y. Zhao. “High-Dimensional, Small-Sample Product Quality Prediction Method Based on MIC-Stacking Ensemble Learning”. In: *Applied Sciences* 12.1. pp. 23. 2021.

[13] P. Refaeilzadeh, L. Tang and H. Liu. “Cross-validation”. In: *Encyclopedia of database systems* 5. pp. 532-538. 2009.

[14] I-C. Yeh. “Modeling of strength of high performance concrete using artificial neural networks”. In: *Cement and Concrete Research* 28.12. pp. 1797-1808. 1998.

[15] I-C. Yeh. “Modeling slump flow of concrete using second-order regressions and artificial neural networks”. In: *Cement and Concrete Composites* 29.6. pp. 474-480. 2007.

[16] F. Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *The Journal of machine Learning research* 12. pp. 2825-2830. 2011.

[17] 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* pp. 785-794. 2016.

[18] G. Ke et al. “Lightgbm: A highly efficient gradient boosting decision tree”. In: *Advances in neural information processing systems* 30. pp. 3149–3157. 2017.

[19] R. Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1. pp. 267-288. 2018.

[20] R. M. Rifkin and R. A. Lippert. “Notes on regularized least squares”. MIT-CSAIL-TR-2007-025. CBCL-268. 2007.

[21] K. P. Murphy “Machine learning: a probabilistic perspective”. MIT press. 2012.

[22] S. J. Kim, K. Koh, M. Lustig, S. Boyd and D. Gorinevsky. “An interior-point method for large-scale ℓ_1-regularized least squares”. In: *IEEE journal of selected topics in signal processing* 1.4. pp. 606-617. 2012.

[23] C. C. Chang “A library for support vector machines”. URL: *ttp://www.csie.ntu.edu.tw/ cjlin/libsvm* 1.4. 2001.

[24] E. Fix and J. L. Hodges. “Discriminatory analysis. Nonparametric discrimination: Consistency properties”. In: *International Statistical Review* 57.3. pp. 238-247. 2001.

[25] C. E. Rasmussen. “Gaussian processes in machine learning”. In: *Summer school on machine learning* pp. 63-71. 2003.

[26] M. Dumont, R. Marée, L. Wehenkel and P. Geurts. “Fast multi-class image annotation with random subwindows and multiple output randomized trees”. In: *International Conference on Computer Vision Theory and Applications (VISAPP).* 2009.

[27] L. Breiman. “Random forests”. In: *Machine learning* 45.1. pp. 5-32. 2009.

[28] P. Geurts, D. Ernst and L. Wehenkel. “Extremely randomized trees”. In: *Machine learning* 63.1. pp. 3-42. 2006.

[29] Y. Freund and R. E. Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of computer and system sciences* 55.1. pp. 119-139. 1997.

[30] J. H. Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* pp. 1189-1232. 2001.

[31] A. Guryanov. “Histogram-based algorithm for building gradient boosting ensembles of piecewise linear decision trees”. In: *International Conference on Analysis of Images, Social Networks and Texts* pp. 39-50. 2001.

[32] T. Head et al. “scikit-optimize: v0.5.2. Version v0.5”. URL: *https://pypi.org/project/scikit-optimize/* 2018.

[33] J. Bergstra, R. Bardenet, Y. Bengio and B. Kégl. “Algorithms for hyper-parameter optimization”. In: *Advances in neural information processing systems* 24. 2011.

[34] R. Mustapha and E. A. Mohamed. “High-performance concrete compressive strength prediction based weighted support vector machines”. In: *International Journal of Engineering Research and Applications* 7.1. pp. 68-75. 2017.

[35] D. C. Feng et al. “Machine learning-based compressive strength prediction for concrete: An adaptive boosting approach”. In: *Construction and Building Materials* 230. 117000. 2020.