sciendo

Civil and Environmental Engineering
Vol. 18, Issue 1, 174-184, DOI: 10.2478/cee-2022-0016

ENSEMBLE TREE MACHINE LEARNING MODELS FOR IMPROVEMENT OF EUROCODE 2 CREEP MODEL PREDICTION

Hikmat DAOU[1,*], Wassim RAPHAEL[1]

[1] Ecole Supérieure d'Ingénieurs de Beyrouth (ESIB), Saint-Joseph University, Beyrouth, Lebanon.
[*] corresponding author: hikmat.daou@net.usj.edu.lb

Abstract

Ensemble tree machine learning models have proved useful for solving poorly understood and complex problems. This paper aims to calibrate the Eurocode 2 creep model by inserting a correction coefficient to the model. The correction coefficient is calculated using ensemble tree (bagging and boosting) models. The results showed that the insertion of the correction coefficient obtained by both the bagging and boosting models into the Eurocode 2 model led to significantly higher prediction accuracy. These approaches may lead to a better performance prediction, thereby reducing the effect of the time-dependent deformation on concrete structures.

Keywords:

Ensemble tree;
Bagging;
Boosting;
Creep;
Eurocode 2.

1 Introduction

As the consumption of concrete continues to increase worldwide, awareness of the behavior of concrete both in the early and late stages of its structural life-cycle has increased. Creep is defined as the time-dependent deformation of concrete under continuous loading. Concrete creep is complex and difficult to measure accurately. In the literature, many complex and practical models have been proposed to predict concrete creep [1, 2]. Designers usually evaluate concrete creep strain using one of two code methods: ACI [1] or Eurocode 2 (EC2) [2]. The American Concrete Institute recommends the ACI model while the Euro-International Committee recommends EC2. Gedam et al. [3] predicted the creep strain of high-performance concrete for up to 900 days using the CEB-FIP 90 model [4], ACI 209 model, B3 model [5], and GL 2000 model [6]. Based on a comparison with the observed experimental data, the CEB-FIP 90 model showed a better prediction than other models. It should be noted that, in addition to the statistical methods, the results of the studies differ because the ranking is determined by the data used to evaluate the models and the selection criteria of the database. Granata et al. [7] studied the effect of concrete properties over time during the construction phase and the lifetime of prestressed concrete girders. They found that the GL 2000 and B3 models gave the highest creep coefficient values and therefore the highest deflection, while the EC2 model gave the smallest values of deflections. The underestimation of the EC2 model can have important implications for identifying construction cambers and assessing service limit states for deformation and cracking. Depending on the structural design, construction material, and service conditions, creep can result in significant displacements, stress redistribution, and prestressing loss [8-9]. Therefore, concrete creep may affect the long-term behavior and serviceability, and even the safety of the structure [10, 11]. Creep in concrete can be affected by many factors such as compressive strength of concrete, type of cement, water-to-cement ratio, aggregate-to-cement ratio, volume-to-surface ratio, age at loading, relative humidity, and sustained load [12].

To obtain a more accurate prediction of creep, many studies aimed to update the models for a better prediction, and many optimization methods were presented [13]. Chen et al. [14] proposed a modified ACI creep model by inserting correction coefficients that are calculated using the particle swarm optimization method to highlight the effect of concrete strength and contents of supplementary cementitious materials. Raphael et al. [15] improved the EC2 creep model by estimating correction factors for different concrete strength categories using different prior distributions using Bayesian

© Author(s) 2022. This work is distributed under the Creative Commons BY 4.0 license (https://creativecommons.org/licenses/by/4.0/).

inference. Moreover, another study [12] has also improved the EC2 model by using Bayesian inference, but with different mixed concrete designs and under various environmental conditions. However, to apply the Bayes theorem, the error must be normally distributed; therefore, the data is chosen or selected and divided to have the error normally distributed. The models in the previous studies were updated using the traditional statistical methods. Indeed, these methods are cumbersome to use, or of limited utility in addressing some problems especially where there are many predictor variables, complex interactions or patterns that exist in data, the variables are not well distributed, etc. In order to overcome these issues, the use of machine learning (ML) techniques for statistical analysis has been proposed by an increasing number of researchers.

ML models can learn the basic behavior of a system on a training dataset without prior knowledge of the nature of the relationships between variables. Ensemble learning is a machine learning paradigm designed to improve reliability, aggregating the results to generate a single prediction. Decision trees, support vector machines, and neural networks are some of the commonly used base learners in ensemble algorithms.

Ensemble tree machine learning models have proved useful for solving poorly understood and complex problems and can be applied to classification and regressions problems. This paper focuses on regression. Dietterich [16] demonstrated significant performance improvements through ensemble methods. Bagging [17] and boosting [18] are the most popular ensemble learning methods due to their theoretical performance guarantees and strong empirical results.

This paper aims to improve creep prediction by introducing a correction coefficient into the EC2 model. Therefore, four ensemble models i.e., Bagged Classification and Regression Trees (Bagged CART), Random Forest (RF), Generalized Boosting Machine (GBM), and Extreme Gradient Boosting Machine (XGBoost) are developed and their hyperparameters are tuned for better performance. The Northwestern University (NU) database is used to build these models.

2 Materials and methods

2.1 Eurocode 2 model

The compliance is the total load-induced deformation (elastic deformation and creep deformation) at age t per unit stress due to a unit uniaxial sustained load since the age of loading t_0. According to Eurocode 2 [2], compliance can be predicted using Eq. (1).

$$J(t,t_0) = \frac{1}{E_{cm}(t_0)} + \frac{\varphi(t,t_0)}{E_{cm28}}, \quad (1)$$

where $J(t,t_0)$ is the compliance, $\varphi(t,t_0)$ is the coefficient of creep, and $E_{cm}(t_0)$ and E_{cm28} are the moduli of elasticity of concrete at the loading age, t_0, and 28 days respectively.

The creep coefficient $\varphi(t,t_0)$ gives the ratio of the creep strain since the start of loading age, t_0, to the elastic strain because of the constant stress applied at a concrete age of 28 days. In this study, the creep coefficient is considered to be updated and calibrated based on the Northwestern University database by inserting correction coefficient, A, into the formula as shown in Eq. (2).

$$\varphi_{upd}(t,t_0) = A \times \varphi(t,t_0). \quad (2)$$

2.2 Northwestern University database

The data is obtained from the Northwestern University (NU) database [19], which was collected in 2010 - 2013 from many journal articles, conference proceedings, and reports. The data used in this study includes 4659 readings of concrete compliance from experiments conducted using different concrete mixtures and under different environmental conditions. All the experiments have been performed under sustained uniaxial compressive normal stress with less than 40 % of the uniaxial compressive strength. In the NU database, the creep is characterized by the compliance function $J(t,t_0)$ which represents the strain at age t caused by sustained uniaxial stress applied at age t_0. Since this study aims to update the EC2 creep coefficient, first the experimental creep coefficient φ_{exp} is calculated using the compliance found in the NU database and the EC2 compliance formula, and therefore the target (correction coefficient) refers to the experimental to EC2 creep coefficient ratio $A = \varphi_{exp}/\varphi_{EC2}$. The potential predictor variables considered in this paper include water-to-cement ratio, aggregate-to-cement ratio, cement content, cement type, compressive strength, loading age,

temperature, relative humidity, volume-to-surface ratio, load over compressive strength at loading age, and time. Fig. 1 represents the path to predict the correction coefficient. The inputs or potential predictor variables with their values range are summarized in Table 1.

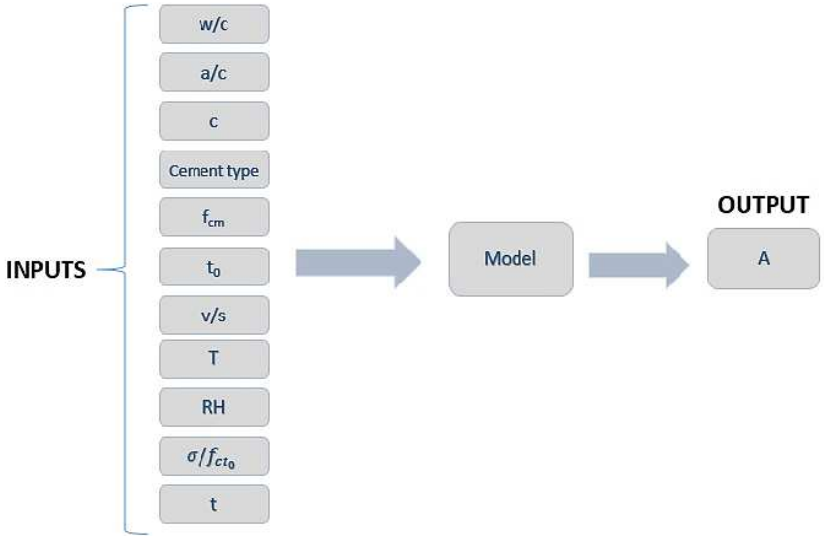


Fig. 1: Diagram showing the inputs, model and output.

Table 1: Values of inputs or potential predictor variables.

Input	Value range
Water-to-cement ratio *w/c*	0.30 - 0.90
Aggregate-to-cement ratio *a/c*	1.22 - 13.24
Cement content *c* [kg/m^3]	214 - 725
Cement type	Sl, R or RS*
28-day compressive strength f_{cm} [MPa]	10.80 - 70.90
Loading age t_0 [days]	1 - 3300
Temperature *T* [°C]	19 - 66
Relative humidity *RH* [%]	20 - 99
Volume-to-surface ratio *v/s* [mm]	5.33 - 128.50
Load over compressive strength at loading age f_{ct0} [MPa]	0.01 - 0.40
Time *t* [days]	1 - 11078

*SL for slow hardening cement, R for normal hardening cement, and RS rapid hardening cement.

2.3 Correction coefficient modeling

In this study, the correction coefficient modeling was conducted using ensemble tree ML methods. The model training and parameters tuning were performed with 10-fold cross-validation using 80 % of the input data. The four machine learning models consist of two Bagging models including Bagged Classification and Regression Trees (Bagged CART) and Random Forest (RF) in addition to two Boosting models including Generalized Boosting Modeling (GBM) and Extreme Gradient Boosting (XGBoost). All models are performed using the R statistical software [20].

2.3.1 Bagged CART

Bagging, also known as bootstrap aggregating, was one of the earliest practitioners of ensemble machine learning algorithms, designed to enhance the accuracy and stability of classification and regression algorithms. Bagging prediction models are a common technique that can fit multiple versions of a prediction model and then combine them into an aggregate prediction [17]. Bagging is a fairly simple algorithm that makes a new prediction by creating a *b* bootstrap copies from the original data, applying a regression algorithm (usually called a base learner) to each bootstrap sample, and averaging the predictions along with the individual base learner.

Bagging works particularly well for unstable and high variance base learner algorithms where the expected results undergo large changes in response to small changes in the training data [21]. Because classification and regression trees (CART) are considered unstable models, bagging can significantly improve accuracy. Bagged CART highly improves regression performance and overfitting, and decreases the prediction variance. Therefore, it is expected that the use of the bagged CART in new applications to improve the model will yield promising results.

Regression-type CART models are a non-parametric procedures for predicting continuous dependent variables with categorical and/or continuous predictor variables. The data is partitioned into nodes based on conditional binary responses to a question involving the predictor variable. CART consistently conduct binary partition for each parameter of the dataset at each builds level until a stopping criterion is met (i.e., the minimum number of data points required to attempt a split before it is forced to create a terminal node, and the maximum number of internal nodes between the root node and the terminal nodes) [22]. During this process, the input data *X* will be partitioned at the split point *s* for splitting variable *j*:

$$R_1(j,s) = \{X|j \leq s\} \; and \; R_2(j,s) = \{X|j > s\}. \tag{3}$$

By repeating the binary partition process, all training datasets will be divided into *m* regions (corresponding to *m* terminal nodes*)* R_1, R_2, … , R_m. Each region R_i corresponds to a prediction response c_i. In order to find the best splitting point *s* for each splitting variable *j*, the mean squared error is used to build up the cost function as follows:

$$J(j,s) = \min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2. \tag{4}$$

For any choice of *j* and *s,* the minimization in Eq. (4) is given by:

$$\widehat{c_1} = \frac{1}{N_1} \sum_{x_i \in R_1(j,s)} y_i^2 \; and \; \widehat{c_2} = \frac{1}{N_2} \sum_{x_i \in R_2(j,s)} y_i^2, \tag{5}$$

where N_1 and N_2 are the number of readings in regions R_1 and R_2, respectively; and y_i is the i-th reading value.

Therefore, CART will search through all possible values of all variables in the dataset for the best partition pair that minimizes the cost function of Eq. (4) [23]. The steps of BaggedCART can be summarized as follows:

- Create a bootstrap sample of the training dataset for each ensemble member.
- Train a CART model on each sample.
- Use statistics (i.e, mean for regression, or mode for classification) to combine the predictions.

The function bagging (), which is based on trees computed by “rpart”, is used in this study. The number of bootstrap samples is controlled by the parameter “nbagg” [24].

2.3.2 Random forests

Random forests are a modification of bagged decision trees that produce a large set of uncorrelated trees to further improve prediction performance. Bagging trees inject random components into the tree-building process by creating multiple trees on bootstrapped copies of the training data and then combining the predictions for all trees. Although this aggregation reduces the variance of the overall procedure and improves prediction performance, simply bagging trees result in tree correlation, which limits the variance reduction effect. Random forests help to reduce tree correlation by adding more randomness to the tree growth process. This can be done by randomly choosing bootstraps for training on and a random sampling of features to use in each partition. The result is a more diverse set of trees that tends to reduce correlations between trees outside the bagged tree and often greatly increases predictive power. In other words, RF are a combination of tree predictors, so each tree depends on the values of a random vector independently sampled with the same distribution for all trees [25].

The procedure of RF can be described as follows:

- Assume the number of cases in the training set *N*.
- Sample randomly the *N* cases.

- Specify the number of variables *m* selected at random out of the input variables. The best split on these *m* variables is used to split the node.
- Each tree is grown to the largest extent possible and there is no pruning.
- Predict new data by aggregating the prediction of all trees (i.e, mean for regression, or mode for classification).

For a fast implementation of RF models, the function ranger is used and the main hyperparameters to be tuned are the number of variables to possibly at in each node (m_{try}), minimal node size (min.node.size), sample with replacement (replace) and fraction of observations to sample (sample.fraction) [26].

2.3.3 Generalized boosting modeling

Generalized Boosting Models (GBM) are very popular machine learning algorithms that have proven to be effective in many areas. GBMs are a combination of two methods: a decision tree algorithm and a boosting method. Boosting frameworks have been generalized for regression problems by linking boosting algorithms to important statistical concepts such as loss functions [27]. GBMs continue to adhere to many decision trees to improve the accuracy of the model accuracy. Using the boosting method, a random subset of all the data is selected for each new tree in the model. The input data for each new tree is weighted, so data that was not well modeled in the old tree is more likely to be picked up by the new tree. This means that the model takes into account errors in predicting whether the first matching tree matches the next, etc. Therefore, the model is constantly trying to improve its accuracy by taking into account the fit of the previously constructed trees.

The package GBM implements the generalized boosted modeling framework. The main hyperparameters to be tuned in GBM are the shrinkage parameter applied to each tree in the expansion (shrinkage), the parameter specifying the maximum depth of each tree (interaction.depth), the parameter specifying the minimum number of observations in the terminal nodes of the trees (n.minobsinnode), and the fraction of the training set observations randomly selected to propose the next tree in the expansion (bag.fraction) [28].

2.3.4 Extreme gradient boosting

Extreme gradient boosting, XGBoost, is an efficient and scalable variant of the Gradient Boosting Machine. It has won several machine learning competitions because of its ease of use, parallel processing, and impressive prediction accuracy. XGBoost is designed to improve speed and performance by introducing regularization parameters to reduce overfitting. Gradient boosted trees use regression trees as weak learners in a sequential learning. This method directly interprets and generalizes the boosting method in terms of numerical optimization of the function space, allowing optimization of arbitrary loss functions [29].

The package xgboost is a simple interface for training XGBoost model. The main parameters to be tuned are the step size of each boosting step (eta), maximum depth of the tree (max. depth), minimum sum of instance weight needed in a child (min_child_weight), subsample ratio of the training instance (subsample), and subsample ratio of columns when constructing each tree (colsample_ by_tree) [30].

2.4 Model validation

Model validation in this study is conducted on 20 % of the data set. The mean absolute error (MAE), the mean absolute percentage error (MAPE) and the root mean squared error (RMSE) are considered to validate the results Eq. (6) and Eq. (8).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \,, \tag{6}$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|, \tag{7}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{8}$$

where y_i is the observed value, $\hat{y}_i$ is the predicted value, and n is the number of predictions.

3. Results and discussion

The correction coefficient *A* (see Eq. (2)) is calculated based on compliance values extracted from the NU database and according to EC2 formulas. Then, the database is explored and checked for outliers. Finally, a total of 4659 values of correction coefficient *A* are used in this study and included as a cohort used to build the ensemble tree models.

The data set is split into two datasets: a training dataset (3727 readings) and a test dataset (932 readings). Noting that the readings from the same experiment are found in one of these two sets (i.e., training or test sets). The models are performed using the R statistical software [20].

3.1 Bagged CART

Bagged CART model suits multiple versions of a prediction model and then combines them into an aggregate prediction. When the number of trees increases, the model accuracy will increase. However, this may come to be computationally intensive as extensive because the number of iterations will increase. Bagged CART model is performed using the function bagging found in ipred library. Fig. 2 shows the error curve for bagging of one to 2000 decision trees.

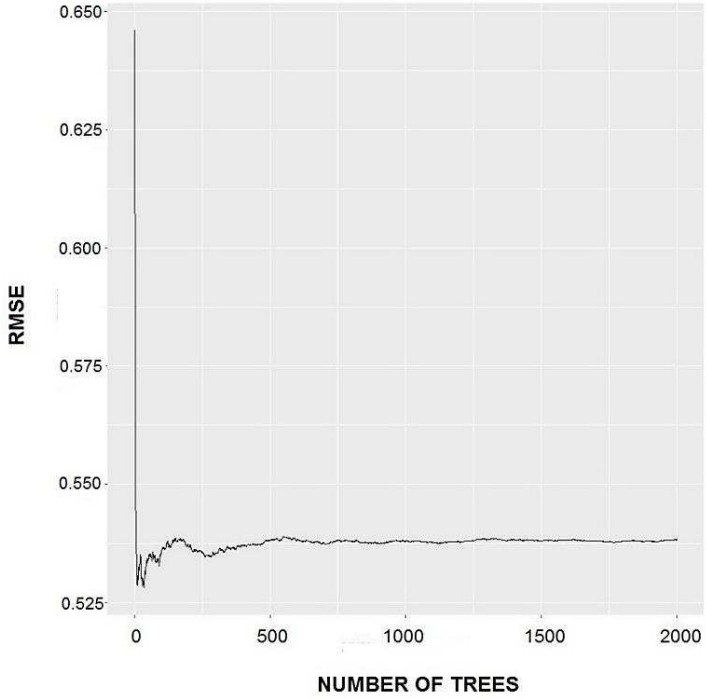


Fig. 2: Error curve for bagging 1 - 2000 deep decision trees.

As shown in Fig. 2, the error is stabilizing with just over 500 trees; therefore, no significant improvement will be achieved by simply bagging more trees. Therefore, in the bagged CART model, 500 bootstrap copies of the training dataset are generated, the base learner (i.e., regression algorithm) is implemented to every bootstrap, and the correction coefficient, *A,* is calculated through averaging the predictions collectively from the individual base learner.

The predictions of the correction coefficient values of the test set are calculated using the function predict and Bagged CART model, and the updated creep coefficients are then computed. Table 2 shows the MAE and RMSE for each model. The results show that, on average, the differences between the true creep coefficient value and the EC2 predictions and the updated EC2 predictions are 0.97 and 0.53, respectively. The MAPE decreases from 53 % to 32 %, and the RMSE decreases from 1.42 to 0.90 after updating the EC2 creep coefficient using Bagged CART. This implies that RMSE decrease 37 % after the implementation of the correction coefficient obtained using the Bagged CART model.

Table 2: Results of MAE, MAPE and RMSE.

	MAE	**MAPE**	**RMSE**
Target value	0	0	0
EC2 model	0.97	0.53	1.42
Bagged CART	0.53	0.32	0.90
RF	0.51	0.28	0.87
GBM	0.53	0.29	0.88
XGBoost	0.51	0.28	0.85

3.2 Random forests

Random forests have several hyperparameters that can be tuned; however, among the most popular machine learning algorithms, prediction accuracy has the least variability when tuned. In this study, the Random forest model is performed using the function ranger found in ranger library. The number of trees in the forest and the number of features to consider for a given partition m_{try} usually have the greatest impact on prediction accuracy. The number of trees should be large enough to stabilize the error rate. The hyperparameter, m_{try}, helps to balance low tree correlation with reasonable predictive power. The complexity of each tree and sampling method have a marginal impact on prediction accuracy, and can also affect computational efficiency. The tree complexity refers to the size of the nodes, maximum depth, maximum number of terminal nodes, and the node size required to allow for further splits. The minimum node size (min.node.size) is the most common hyperparameter for managing tree complexity. For the sampling method, the sample size (sample.fraction) must be adjusted and the sample must be specified with or without replacement. When there are many categorical features with different numbers of levels, sampling with replacement can lead to biased variable split selection. In order to tune these hyperparameters, a full Cartesian grid search is used to assess each combination of hyperparameters of interest, Table 3. Then, a search across the 120 combinations settings is used, and finally, the best hyperparameters combination is obtained.

Table 3: Values of hyperparameters adjusted in the grid search for RF model.

Hyperparameter	**Tested value**				
m_{try}	1	2	3	**4**	5
Min.nod.size	**1**	3	5	10	
Replace	yes	**no**			
Sample.fraction	0.50	0.63	**0.80**		

The best model refers to the combination of (m_{try} = 4; min.node.size = 1; replace = no; sample.fraction = 0.80). Therefore, m_{try} of 4 is almost sufficient and a smaller node size (deeper trees) performs best. In addition, sampling by 80 % without replacement leads to better results. In fact, sampling less than 100 % helps to further decorrelate trees by adding randomness to the procedure, and sampling without replacement improves performance because the data has many unbalanced categorical features with high cardinality.

The predictions of the correction coefficient values of the test set are calculated using the function predict() and RF model, and the updated creep coefficients are then computed. As shown in Table 2, the MAE, MAPE and RMSE decrease to 0.51, 28 % and 0.87, respectively. Therefore, a significant improvement in creep coefficient prediction is encountered if the correction coefficient obtained from the RF model is implemented in the EC2 model.

3.3 Generalized boosting model

GBM iteratively improves the regression tree, and each model in the sequence slightly improves the performance of the previous model. This improvement is primarily achieved by focusing on the training data rows that had the largest errors or residuals in the previous tree. GBM is performed using the function *gbm()* found in the gbm library.

GBMs are known to be very flexible and have several tuning parameters. The problem is that tuning and finding the optimal combination of GBM hyperparameters can take a long time. Simple GBM has two categories of hyperparameters (boosting and tree-specific hyperparameters). The first

category mainly contains the number of trees and the learning rate. The optimal number of trees represents the number of trees that minimize the cross-validation loss function of interest. The learning rate, also known as shrinkage, determines the contribution of each tree to the outcome and controls the speed at which the algorithm moves along gradient descent. The second category includes the tree depth (interaction.depth), minimum number of observations in terminal nodes (n.minobsinnode), and subsampling (bag.fraction). The depth of a tree determines the depth of an individual tree, and the minimum number of observations in terminal nodes determines the complexity of each tree. Subsampling controls whether a subset of the available training observations is used.

In GBM, the parameters usually need to be tuned. For that, a grid search is performed to iterate over each combination of hyperparameters values, see Table 4 in order to know which combination works best. Therefore, a grid search is created to search across 81 models.

Table 4: Values of hyperparameters adjusted in the grid search for GBM.

Hyperparameter	**Tested value**		
Shrinkage	0.01	0.1	**0.3**
Interaction.depth	1	3	**5**
N.minobsinnode	5	10	**15**
Bag.fraction	0.65	0.80	**1**

The best model refers to the combination of (shrinkage = 0.3; interaction.depth = 5; n.minobsinnode = 15; bag.fraction = 1). The top model uses a shrinkage of 0.3. In other words, the gradient descent method works best when the incremental steps are larger. Additionally, a larger value of interaction.depth indicates that there are likely to be significant interactions that deeper trees can capture. However, larger nodes can also capture pockets of unique feature-correction coefficient point instances. The best model uses all the available training observations. The optimal number of trees obtained is 2845 trees.

After predicting the value of the correction coefficient using the function predict and GBM model, and implementing it in the EC2 model, the MAE, MAPE and RMSE decrease to 0.53, 29 % and 0.88, respectively, see Table 2. Therefore, a significant improvement in creep coefficient prediction is noticed after the implementation of the correction coefficient obtained using the GBM in the EC2 model.

3.4 Extreme boosting model

XGBoost is a recently developed gradient boosting machine tree-based model. GBMs continue to improve to minimize any errors; therefore, the XGBoost model uses a 10-fold cross-validation to avoid overemphasizing outliers and overfitting. XGBoost model is performed using the function xgboost found in xgboost library.

The most common parameters for tuning an XGBoost model are learning rate (eta), tree depth (max_depth), minimum node size (min_child_weight), training data-to-sample ratio for each tree (subsample), and columns-to-sample ratio from for each tree (colsample_by_trees). A large search grid of 576 different hyperparameters combinations is performed using the values of hyperparameters listed in Table 5. The model loops through each hyperparameter combination to know which combination works best.

Table 5: Values of hyperparameters adjusted in the grid search.

Hyperparameter	**Tested value**			
Eta	0.01	**0.05**	0.10	0.30
Max_depth	1	3	5	**7**
Min_child_weight	1	**3**	5	7
Subsample	0.65	0.80	**1**	
Colsample_by_trees	**0.8**	0.90	1	

XGboost uses shrinkage (or learning rate) and column subsampling for gradient boosting trees to further reduce overfitting. Thus, it adds the overfitting treatment techniques introduced in Adaboost

(learning rate) and random forests (subsampling of columns and feature) to the regularization parameters used in stochastic gradient descent models.

The best model refers to the combination of (eta = 0.05; max_depth = 7; min_child_weight = 3; subsample = 1; colsample_by_trees = 0.8). The best model uses a learning rate of 0.05; thus, to reduce the risk of overfitting, it is better that the incremental steps down the gradient be smaller. The depth of trees in the top model is seven, which means that there are some important interactions that deeper trees can capture. The minimum node size of 3 allows the model to capture pockets of unique feature-correction coefficient point instances. Moreover, the top model uses a column sampling subsampling of 80 %, which means that adding a stochastic component can be useful in finding the global optimum of the loss function gradient and reducing the chances of being stuck in local minima, plateaus, or other irregular terrain of the loss function.

XGBoost model with the best combination is performed, and the predictions of the correction coefficient are calculated using the function predict. Finally, the updated creep coefficient is computed for the test set, and the statistical indicators are shown in Table 2. The MAE, MAPE and RMSE decrease by 47 %, 46 % and 40 %, respectively, after the implementation of the correction coefficient obtained using the XGBoost model.

As shown in Table 2, there is a significant improvement in the creep coefficient prediction after implementing the correction coefficient into the EC2 model. As the results of the four models are pretty similar (see Table 2), the ensemble tree models proved useful for calculating the correction coefficient, and the implementation of this coefficient that is computed by any of these models will greatly improve creep prediction. However, RF and XGBoost models give the best predictions according to the MAE and MAPE while XGBoost model performs best for RMSE. For future applications, it is recommended to implement the correction coefficient into the Eurocode 2 which is computed using the XGBoost model with the following set of hyperparameters (eta = 0.05; max_depth = 7; min_child_ weight = 3; subsample = 1; colsample_by_trees = 0.8).

4 Conclusion

Creep, which is the time-dependent deformation of concrete under continuous loading, can result in stress distribution, service problems, and prestress loss. This paper aims to improve creep prediction by adding correction coefficients to the EC2 model. It is reported that most ensemble machine learning methods give more accurate predictions. Therefore, bagging and boosting based machine learning models were developed to calculate the correction coefficients to be implemented in the EC2 model. The models were built using the Northwestern University (NU) database, and the correction coefficients were predicted based on eleven variables. These variables include water-to-cement ratio, cement type, cement content, aggregate-to-cement ratio, compressive strength, loading age, volume-to-surface ratio, load to strength at loading age ratio, relative humidity, temperature, and time.

Bagging was one of the first ensemble algorithms studied by machine learning experts. It is designed to improve the accuracy and stability of classification and regression algorithms. Bagged CART and Random Forest models were performed. In Bagged CART, the optimal number of decision trees was determined and the model was then defined. The hyperparameters of the RF model were tuned by performing a search across a grid search; therefore, the best model was performed.

Boosting is a common algorithm for building an ensemble of decision trees. A generalized boosting algorithm, GBM, combines two methods: a decision tree algorithm and a boosting method. Extreme gradient boosting, XGBoost, which is an efficient and scalable version of the gradient boosting machine, was designed to improve speed and performance by injecting regularization parameters to reduce overfitting. The parameters of these two models (GBM and XGBoost) were adjusted to give the best results.

The mean absolute error (MAE) and root mean square of errors (RMSE) were used to check the validity of the models. Application of the four models (Bagged CART, RF, GBM and XGBoost) to the test set showed a good discrimination ability and accuracy. Therefore, the bagging and boosting models resulted in a significant improvement of the EC2 model in creep prediction; however, XGBoost model gave the best results where the MAE, MAPE and RMSE have decreased by 45 %, 46 % and 39 %, respectively.

Calculating the correction coefficient using the obtained machine learning models with their optimal set of hyperparameters and implementing it into the Eurocode 2 will lead to a better performance prediction of creep, thereby reducing the effect of time-dependent deformations on

concrete structures. For future works, others machine learning models such as deep neural networks or deep belief networks may be developed to predict creep of concrete.

References

[1] ACI COMMITTE 209: Guide for Modeling and Calculating Shrinkage and Creep in Concrete Structures (ACI 209.2R-08), 2008.
[2] CEN:EN 1992-1-1, Eurocode 2: Design of concrete structures - Part 1-1 : General rules and rules for buildings, 2004.
[3] GEDAM, B. A. - UPADHYAY, A. - BHANDARI, N. M.: An apt material model to predict creep and shrinkage behaviour of HPC concrete. Sustain Constr Mater Technol, Vol. August, 2013.
[4] CEB-FIP, MC90: Design of concrete structures. CEB-FIP Model Code 1990, British Standard Institution, 1993.
[5] BAZANT, Z. P. – BAWEJA, S.: Creep and shrinkage prediction model for analysis and design of concrete structures: Model B3. ACI Spec Publ, Vol. 194, 2000, pp. 1-84.
[6] GARDNER, N. J – LOCKMAN, M. J..: Design provisions for drying shrinkage and creep of normal-strength concrete. Materials Journal, Vol. 98, 2001.
[7] GRANATA, M. F. - MARGIOTTA, P. - ARICI, M.: A Simplified Procedure for Evaluating the Effects of Creep and Shrinkage on Prestressed Concrete Girder Bridges and the Application of European and North American Prediction Models. J Bridg Eng, Vol. 18, Iss. 12, 2013, pp. 128-197.
[8] KRALOVANEC, J. - MORAVCIK, M. - JOST, J.: Analysis of Prestressing in Precast Prestressed Concrete Beams. Civil and Environmental Engineering, Vol. 17, Iss. 1, 2021, pp. 184-191.
[9] RAPHAEL, W. - ZGHEIB, E. - CHATEAUNEUF, A.: Experimental investigations and sensitivity analysis to explain the large creep of concrete deformations in the bridge of Cheviré. Case Stud Constr Mater, Vol. 9, 2018, e00176.
[10] RAPHAEL, W. - FADDOUL, R. - GEARA, F. - CHATEAUNEUF, A.: Improvements to the Eurocode 2 shrinkage model for concrete using a large experimental database. Struct Concr, Vol. 13, Iss. 3, 2012, pp. 174-181.
[11] JERGA, J. - KRAJCI, Ľ.: Damage in Concrete and its Detection by Use of Stress-Volumetric Strain Diagram. Civil and Environmental Engineering, Vol. 10, Iss. 1, 2014, pp. 16-25.
[12] DAOU, H. - RAPHAEL, W.: A Bayesian regression framework for concrete creep prediction improvement: application to Eurocode 2 model. Res Eng Struct Mater, 2021, pp. 1-19.
[13] FADDOUL, R. - RAPHAEL, W. - CHATEAUNEUF, A.: Maintenance optimization of series systems subject to reliability constraints. Reliab Eng Syst Saf, Vol. 180, 2018, pp. 179-188.
[14] CHEN, P. - ZHENG, W. - WANG, Y. - CHANG, W.: Creep model of high-strength concrete containing supplementary cementitious materials. Constr Build Mater, Vol. 202, 2019, pp. 494-506, https://doi.org/ 10.1016/j.conbuildmat.2019.01.005.
[15] RAPHAEL, W. - FADDOUL, R. - SELOUAN, A. - CHATEAUNEUF, A.: Information-based formulation for Bayesian updating of the Eurocode 2 creep model. Struct Concr, Vol. 10, Iss. 2, 2009.
[16] DIETTERICH, T.: Ensemble Methods in Machine Learning. Int Work Mult Classif Syst, 2000, pp. 1-15.
[17] BREIMAN, L.: Bagging predictors. Mach Learn, Vol. 24, 1996, pp. 123-140.
[18] FREUND, Y. - SCHAPIRE, R. E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. J Comput Syst Sci, Vol. 55, Iss. 1, 1997, pp. 119-139, https://doi.org/ 10.1006/jcss.1997.1504.
[19] BAZANT, Z. – LI, G. H.: Comprehensive database on concrete creep and shrinkage. ACI Mater J, Vol. 105, 2008, pp. 635-637.
[20] R CORE TEAM.: R: A language and environment for statistical computing. R foundation for statistical computing, Vienna, Austria, 2021.
[21] DIETTERICH, T.: An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. Mach Learn, Vol. 40, 2000, pp. 139-157.
[22] THERNEAU, T. - ATKINSON, B.: rpart: Recursive Partitioning and Regression Trees. R package version 4, 2019, pp. 1-15.
[23] LIANG, M. - CHANG, Z. - WAN, Z. - GAN, Y. - SCHLANGEN, E. - SAVIJA, B.: Interpretable Ensemble- Machine-Learning Models For Predicting Creep Behavior of Concrete. Cem Concr Compos, Vol. 125, 2022.
[24] PETERS, A. - HOTHORN, T.: ipred: Improved Predictors. R package version 0.9-9, 2019.
[25] BREIMAN, L.: Random forests. Mach Learn, Vol. 45, 2001, pp. 5-32.
[26] WRIGHT, M. N. - ZIEGLER, A.: ranger: A Fast Implementation of Random Forests for High

Dimensional Data in C++ and R. J. Stat. Softw., Vol. 77, 2017, pp. 1-17.
[27] FRIEDMAN, J. H.: Greedy Function Approximation: A Gradient Boosting Machine. Ann Stat, Vol. 29, Iss. 5, 2001, p. 1189–232, www.jstor.org/stable/2699986.
[28] GREENWELL, B. - BOEHMKE, B. - CUNNINGHAM, J.: gbm: Generalized Boosted Regression Models. R package version 2.1.8, 2020.
[29] CHEN, T. - GUESTRIN, C.: XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining New York, NY, USA: Association for Computing Machinery, 2016, pp. 785–94, https://doi.org/10.1145/2939672.2939785
[30] CHEN, T. - HE, T. - BENESTY, M. - KHOTILOVICH, V. - TANG, Y. - CHO, H. - CHEN, K. - MITCHELL, R. - CANO, I. - ZHOU, T. - LI, M. - XIE, J. - LIN, M. - GENG, Y. - LI, Y.: xgboost: Extreme Gradient Boosting. R package version 1.5.0.2, 2021. .