Computers and Concrete, *Vol. 21, No. 4 (2018) 407-417*
DOI: https://doi.org/10.12989/cac.2018.21.4.407

Prediction of concrete compressive strength using non-destructive test results

Hamit Erdal*[1], Mürsel Erdal[2], Osman Şimşek[2] and Halil İbrahim Erdal[3]

[1]*Institute of Social Sciences, Atatürk University, 25240, Erzurum, Turkey*
[2]*Technology Faculty, Department of Civil Engineering, Gazi University, 06500 Teknikokullar, Ankara, Turkey*
[3]*Turkish Cooperation and Coordination Agency (TİKA), Ataturk Bulvarı, No. 15, Ulus, Ankara, Turkey*

(Received July 3, 2017, Revised November 21, 2017, Accepted December 13, 2017)

Abstract. Concrete which is a composite material is one of the most important construction materials. Compressive strength is a commonly used parameter for the assessment of concrete quality. Accurate prediction of concrete compressive strength is an important issue. In this study, we utilized an experimental procedure for the assessment of concrete quality. Firstly, the concrete mix was prepared according to C 20 type concrete, and slump of fresh concrete was about 20 cm. After the placement of fresh concrete to formworks, compaction was achieved using a vibrating screed. After 28 day period, a total of 100 core samples having 75 mm diameter were extracted. On the core samples pulse velocity determination tests and compressive strength tests were performed. Besides, Windsor probe penetration tests and Schmidt hammer tests were also performed. After setting up the data set, twelve artificial intelligence (AI) models compared for predicting the concrete compressive strength. These models can be divided into three categories (i) Functions (i.e., Linear Regression, Simple Linear Regression, Multilayer Perceptron, Support Vector Regression), (ii) Lazy-Learning Algorithms (i.e., IBk Linear NN Search, KStar, Locally Weighted Learning) (iii) Tree-Based Learning Algorithms (i.e., Decision Stump, Model Trees Regression, Random Forest, Random Tree, Reduced Error Pruning Tree). Four evaluation processes, four validation implements (i.e., 10-fold cross validation, 5-fold cross validation, 10% split sample validation & 20% split sample validation) are used to examine the performance of predictive models. This study shows that machine learning regression techniques are promising tools for predicting compressive strength of concrete.

Keywords: concrete; compressive strength; machine learning regression; non-destructive testing

1. Introduction

Concrete which is a composite material is one of the most important construction materials. Compressive strength is a commonly used parameter for the assessment of concrete quality. Although destructive methods of compressive strength determination in which cube or cylindrical samples prepared from fresh concrete or core samples extracted from structural concrete members are the most accurate ways, they have their own shortcomings.

Cube or cylindrical samples casted from fresh concrete may not be identical to in-situ concrete because of curing and placement differences. Coring process is time consuming, uneconomical and this process may damage the structural member (Neville 1993, Mehta and Monterio 2006). Because of these disadvantages of destructive test methods, nondestructive test methods are also preferred. Schmidt hammer test in which surface hardness is indirectly measured is widely used for compressive strength estimation and it has the advantage of being economical, fast and non-destructive

However this test only reflects the surface properties of concrete and it may not accurately estimate the internal strength (Mehta and Monterio 2006, Erdal and Simsek 2006, Aydin and Saribiyik 2010).

Another popular nondestructive test method for the determination of compressive strength of concrete is the pulse velocity test. In this method the velocity of sound waves transmitted though the concrete specimen is measured. This velocity is dependent on the dynamic stiffness of the concrete (Kewalramani and Gupta 2006, Solis-Carcano and Moreno 2008, Mielentz 2008, Trtnik *et al.* 2009, Antonaci *et al.* 2010, Breysse 2012).

Accurate prediction of concrete compressive strength is an important issue. Development of prediction models for this issue leads to saving time, costs, equipment and allow making a successful mixture. In this section a brief literature review about the use of statistical and machine learning techniques for facing this problem is aimed. Researchers firstly have employed classical regression techniques for this purpose (Galan 1967, Domone and Soutsos 1994, Yeh 1998).

In addition to these popular non-destructive test methods, a relatively new technique called as Windsor probe penetration test is also utilized for the estimation of compressive strength. In this method, compressive strength is indirectly estimated using the penetration of a probe in to the concrete which is charged with explosives. Lesser the depth of penetration of the probe means the higher the compressive strength of concrete (Mehta and Monterio 2006, Windsor Probe Test System Inc. 1994, Erdal 2002, Baykan *et al.* 2017).

Many empirical equations (i.e., Single-variable, multi-variable) based on regression technique in which the results of nondestructive tests are used, were developed for the

∗Corresponding author, Ph.D.
E-mail: hamit_erdal@hotmail.com

ISSN: 1598-8198 (Print), 1598-818X (Online)

estimation of compressive strength of concrete. Users of nondestructive tests are facing with the problem of choosing the empirical equation which has the highest estimation performance (Ramyar and Kol 1996, Kheder 1998, Qasrawi 2000, Erdal 2009).

Empirical modeling using regression can have remarkable disadvantages. For performing regression analyses, the structure of the model should be pre-defined by a linear or nonlinear equation (Mousavi *et al.* 2010). To choose optimum equation form is a challenging task.

Predicting a real value, which is known as regression, is one of the most common problem among the machine learning community. For this reason, machine learning algorithms are used to control response of a system for predicting a numeric or real-valued target feature. Many real-life problems can be solved as regression problems, and evaluated using machine learning approaches to develop predictive models (Tufekci 2014).

Machine learning regression techniques are useful to develop predictive models but their use requires insight into the learning problem formulation, selection of appropriate learning methods, and evaluation of modeling results to achieve the stated goal of the modeling activity. This paper compares the predictive accuracy of twelve selected machine learning regression techniques using performance statics. The selected machine learning techniques are (i) Functions (i.e., Linear Regression, Simple Linear Regression, Multilayer Perceptron, Support Vector Regression), (ii) Lazy-Learning Algorithms (i.e., IBk Linear NN Search, KStar, Locally Weighted Learning) (iii) Tree-Based Learning Algorithms (i.e., Decision Stump, Model Trees Regression, Random Forest, Random Forest, Reduced Error Pruning Tree). Functions contain algorithms, which are based on the mathematical models. Lazy-learning algorithms delay dealing with training data until a query is answered. They store the training data in memory and find relevant data in the database to answer a particular query. Tree-based learning algorithms are used for making predictions via a tree structure. Leaves of the tree structures illustrate classifications and branches of the tree structures denote conjunctions of features. The detailed description of machine learning regression approaches and the study approach are presented in the following sections.

Over the last years, machine learning has attracted much attention in academic fields for handling civil engineering problems. The machine learning techniques are powerful tools for prediction models with numerous applications including Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Linear Regression Analysis, and others. ANNs are the most widely used machine learning technique in order to address the prediction of concrete strength among them (Yeh 2007).

Recent researches are performed for the usability of ANN in the civil engineering and especially for the concrete technology. Hola and Schabowicz (2005) utilized ANN's for the determination of concrete compressive strength. They suggested that ANN has a good predictive capacity. Topcu and Saridemir (2008) used ANN and Fuzzy Logic for the determination of compressive strength of fly ash added concretes. Topcu and Saridemir (2008) concluded that both ANN and Fuzzy Logic methods have high predictive performance. Altun *et al.* (2008) used ANN and multiple linear regression techniques for the estimation of compressive strength of steel fiber reinforced concrete, their findings then multiple linear regression technique. Subasi (2009) developed on ANN for the estimation of mechanical properties of fly ash added cement paste and concluded that ANN has better performance than multiple linear regression technique.

Atici (2011) applied multiple regression analysis and an ANN to predict the concrete compressive strength. The obtained results indicated that the ANN models performed better than multiple regression analysis models. Saridemir *et al.* (2009) proposed an ANN and fuzzy logic models for prediction of long-term effects of ground granulated blast furnace slag on concrete compressive strength under wet curing conditions. Rajasekaran and Amalraj (2002) and Rajasekaran *et al.* (2002) developed prediction models for the strength of concrete mixes using sequential learning neural network (SLNN). Rajasekaran and Lavanya (2007) employed wavelet neural network (WNN) method to assess the compressive strength. Sobhani *et al.* (2010) used regression model, ANNs, and adaptive fuzzy and neural systems to predict the compressive strength of slump-free concrete. Lee (2003) developed single and multiple ANN architectures for predicting of concrete strength, while Dias and Pooliyadda (2001) utilized back propagation NNs for predicting the strength and slump of concrete.

Genetic programming (GP) is another branch of the machine learning methods. There have been some efforts aiming to apply GP to civil engineering issues. GPs have been utilized to derive simplified models for civil engineering problems. Mousavi *et al.* (2010) have recently derived prediction model for the compressive strength of concrete mixes using a combined algorithm of GP and orthogonal least squares (OLS), called GP/OLS. They formulated the compressive strength in terms of ratio of water and superplasticizer summation to binder, coarse to fine aggregate content and age of specimens.

In the field of civil engineering, some authors have focused on hybridizing prediction techniques. Many of them have reported on hybrid techniques which are able to predict the compressive strength to a high degree of accuracy (Cheng 2012).

Fazel Zarandi *et al.* (2008) proposed a fuzzy polynomial neural network (FPNN) that combined fuzzy neural networks (FNNs) and polynomial neural networks (PNNs) to predict the compressive strength, while Yeh and Lien (2009) applied a genetic operation tree (GOT), which combines an operation tree and a genetic algorithm to automatically produce self-organized equations for the prediction. Another research applied by Cheng and Wu (2009) was The Evolutionary Support Vector Machine Inference Model (ESIM), one hybridization technique, uses a fast messy Genetic Algorithm (fmGA) and SVM to search simultaneously for the best SVM parameters within an optimized legal model. Gupta *et al.* (2006) applied a neural-fuzzy inference system for predicting the compressive strength.

The rest of this article is organized into four additional sections. In Section 2, the experimental study is presented briefly; the Section 3 is devoted to methods; in the 4th

Table 1 Amount of materials used for fresh concrete production (1m³)

Material	Amount
Crashed coarse aggregate (16-25 mm)	334 kg
Crushed medium aggregate (4-16 mm)	632 kg
Crushed fine aggregate (0-4 mm)	761 kg
Cement (PC 42.5)	426 kg
Tap water	190 lt

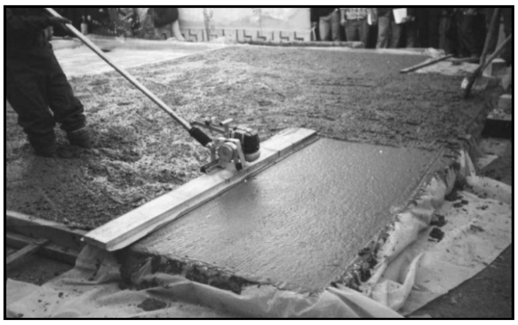
Fig. 1 Compaction of concrete with vibrating screed

Section findings are summarized. Lastly, the 5th section includes the conclusions and final assessments.

2. Experimental studies

Experimental studies consist of sample preparation, curing, application of nondestructive tests, coring, compressive strength determination by destructive tests stages.

Table 1, presents the grain size distributions of aggregate, cement and tap water amount for 1 m³ fresh concrete.

Concrete mix was prepared according to C 20 type concrete, slump of fresh concrete was about 20 cm, and the thickness of the concrete was 15 cm. Prior to concrete placement a polyethylene membrane was laid to the bottom surface in order to apply vacuum properly and to prevent fractures due to ground surface. After the placement of fresh concrete to formworks, compaction was achieved using a vibrating screed (Fig. 1).

Soon after the compaction stage, vacuum sheet was placed to concrete surface. Duration of vacuum application was 34 min to first formwork, 17 min to second formwork. Vacuum was not applied to third formwork.

After 28 day period, a total of 100 core samples having 75 mm diameter were extracted according to ASTM C 42/C 42M (1999) (Fig. 2). Length to diameter ratio of core samples was about 2.

Compressive strength values of core samples were determined using a stress/or strain controlled compression machine according to ASTM C 39 (2001). Windsor probe penetration tests (ASTM C803, 1999) (Fig. 3), Schmidt hammer tests (ASTM C805, 1997) were performed directly on concrete slabs prior to coring. Whereas pulse velocity tests (ASTM C597, 1998) were performed on core samples. The test results are demonstrated in Table 2.

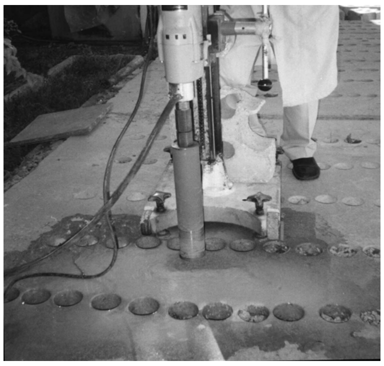
Fig. 2 Extraction of core samples for testing

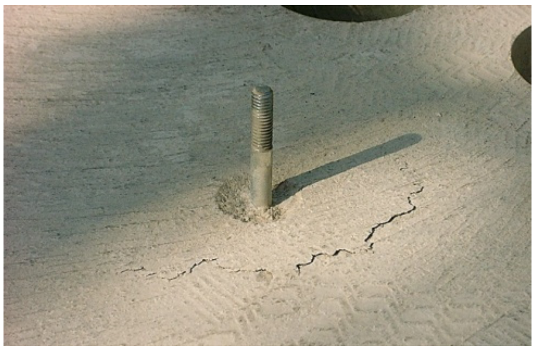
Fig. 3 Penetrated probe

Table 2 The compressive strength values of concrete determined by different methods

Method	Number of sample	Mean (N/mm^2)	Std. Dev.	Min.	Max.
Windsor probe penetration test	20	32.56	5.1656	23.73	38.93
Schmidt hammer test	20	35.65	3.0515	29.90	43.10
Pulse velocity tests	20	32.94	1.6757	30.31	35.80

3. Machine learning regression methods

Machine learning (ML) regression algorithms predict an unknown dependency between the inputs and output from a dataset (Tufekci 2014). Table 3, shows a list of the ML regression methods, which are used in this study. Most of these regression methods have been commonly used for modeling many real-life regression problems. These methods are divided into three categories such as Functions, Lazy-learning Algorithms, and Tree-based Learning Algorithms, stated by The Waikato Environment for Knowledge Analysis (WEKA) platform. In this study version 3.7.11 of WEKA was used. All the machine learning regression algorithms are used with their default parameter settings, as defined in WEKA 3.7.11, to reduce the danger of over fitting due to excessive parameter tuning. Functions contain algorithms, which are based on the mathematical models. Lazy-learning algorithms hold-up handling with training data until a query is answered. These algorithms accumulate the training data in memory and find related data in the database to answer a particular query. Tree-based learning algorithms are used for making predictions via a

Table 3 Regression methods used in this study

Categories	Method	Abbreviation
Functions	Linear Regression	LR
	Simple Linear Regression	SLR
	Multilayer Perceptron	MLP
	Support Vector Regression	SMOreg
Lazy-learning algorithms	IBk Linear NN Search	IBk
	KStar	*K**
	Locally Weighted Learning	LWL
Tree-based learning algorithms	DecisionStump	DecisionStump
	Model Trees Regression	M5P
	RandomForest	RandomForest
	RandomTree	RandomTree
	Reduced Error Pruning Tree	REPTree

tree structure. Leaves of the tree structures exemplify classifications and branches of the tree structures indicate conjunctions of features. The brief summary of the methods, used in this study, are presented in Table 3.

3.1 Linear Regression (LR)

The LR presents a mathematical model of the relationship between a dependent variable and one or more independent variables. LR handles with the weighted instances to form a prediction model. If there is some linear dependency among the data, LR may create a best predictive model, which is a linear regression equation to predict the outputs (x) for a set of input attributes a_1, $a_2,\ldots,a_k$. In Eq. (1), w_0, $w_1,\ldots,w_k$, are the weights respectively of each input attribute, where w_1 is the weight of a_1 and a_0 is always considered as the constant 1. An equation takes the form

$$x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k \tag{1}$$

The weights must be selected to minimize the difference between the actual and predicted output values. In Eq. (2), the predicted output value for the first instance of a training dataset is obtained as

$$w_0 + w_1 a_1^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)} \tag{2}$$

After the predicted outputs for all instances are obtained, the weights are reassigned so as to minimize the sum of squared differences between the actual and predicted outcome as in Eq. (3) (Witten and Frank 2005, Erdal 2013). So the aim of the weight update process is to minimize the sum of the squared differences between the observed output for the I^{th} training instances ($X^{(i)}$) and the predicted outcome for that training instance calculated from the linear regression equation (Erdal 2013).

$$\sum_{i=1}^{n} (x^i - \sum_{j=0}^{k} w_j a_j^{(1)}) \tag{3}$$

3.2 Simple Linear Regression (SLR)

The SLR generates a regression model, which has lowest squared error as the final model among each parameter of the model. This model fits straight models between each input attribute (a_1 and a_0) and output (x) as in Eq. (4), in which the values of w and w_0, which are the weight of a_1 and a_0, are predicted by the method of least squares. In Eq. (4), a_0 is assumed as the constant 1.

$$x = w_0 + w a_1 \tag{4}$$

The predictive model is selected by minimizing the squared error, which is the difference between the actual values and the predicted values (Ekinci *et al.* 2011).

3.3 Multi-Layer Perceptron (MLP)

The MLP is a feed forward artificial neural network (ANN) model that consists of neurons with substantially weighted interconnections, where signals always travel to the direction of the output layer. These neurons are mapped as sets of input data onto a set of suitable outputs with hidden layers (Erdal *et al.* 2013). The input signals are sent by the input layer to the hidden layer without carrying out any operations. Then the hidden and output layers multiply the input signals by a set of weights, and either linearly/nonlinearly transforms results into output values. The connection between units in following layers has an associated weight. These weights are optimized to compute reasonable accuracy of prediction. A typical MLP with one hidden layer can be mathematically defined in Eqs. (5)-(9) as below (Haykin 1999, Erdal 2015):

$$u_j = \sum_{i=1}^{N_{input}} X_i a_{ij} + a_{0j} \tag{5}$$

Eq. (5) defines summing products of the inputs (X_i) and weight vectors (a_{ij}) and a bias term of hidden layer (a_{0j}). In Eq. (6), the outputs of hidden layer (Z_j) are obtained as transforming this sum, that is defined in Eq. (5), by using transfer function (activation function) g.

$$Z_j = g(u_j) \tag{6}$$

The most commonly used transfer function is sigmoid function, that is defined in Eq. (7) for input x. The hidden and output layers are based on this sigmoid function.

$$g(x) = sigmoid(x) = \frac{1}{(1+e^{-x})} \tag{7}$$

Eq. (8) defines summing products of hidden layer's outputs (Z_j) and weight vectors (b_{jk}) and bias term of output layer (b_{0k}).

$$v_k = \sum_{j=1}^{N_{hidden}} Z_j b_{jk} + b_{0k} \tag{8}$$

In Eq. (9), the outputs of the output layer (Y_k) are obtained by transforming this sum, that is calculated in Eq. (8), using sigmoid function g, which is defined in Eq. (7) (Betrie 2013, Namli *et al.* 2016).

$$Y_k = g(v_k) \tag{9}$$

3.4 Support Vector Regression (SMOreg)

The basic idea in Support Vector Regression (SMOreg) is to map the input data *x* into a higher dimensional feature space by nonlinear mapping to solve a linear regression problem in this feature space (Wang and Xu 2004). This transformation (mapping) can be done using a kernel function. The most common kernel functions are linear kernel, polynomial kernel, Gaussian (RBF) kernel, and sigmoid (MLP) kernel (Erdal and Karakurt 2013). In SMOreg method, the regression function is approximated by the following function (10)

$$y = \sum_{i=1}^{I} w_i \phi_i(x) + b \tag{10}$$

where $\{\phi_i(x)\}_{i=1}^{I}$ are the features of inputs, $\{w_i\}_{i=1}^{I}$ and *b* are coefficients. The coefficients are predicted by minimizing the regularized risk function

$$R(C) = C\frac{1}{I}\sum_{i=1}^{I} L_\varepsilon(d_i, y_i) + \frac{1}{2}\left\|\vec{w}\right\|^2 \tag{11}$$

where If $|d-y| \geq \varepsilon$ then $L_\varepsilon(d,y)=|d-y|-\varepsilon$, otherwise $L_\varepsilon(d,y)=0$ and ε is a prescribed parameter (Ekinci *et al.* 2011, Yaprakli and Erdal 2015, Turkan *et al.* 2016).

3.5 IBk Linear NN Search (IBk)

The IBk instance-based learning that works as a *k*-nearest-neighbor classifier, which is the most commonly used instance-based or lazy method for both classification and regression problems. In this paper, it is used for a regression problem. The main assumption behind this algorithm is that the closest instances to the query point have similar target values to the query. The algorithm normalizes attributes by default and can do distance weighting. A variety of different search algorithms are used to speed up the task of finding the nearest neighbors (Jiawei and Kamber 2001).

3.6 KStar (K)*

KStar method is also an instance-based classifier used for regression (Cleary and Trigg 1995). The KStar algorithm uses entropic measure, based on probability of transforming instance into another by randomly choosing between all possible transformations. Using entropy as appraise of distance has numerous utility. A consistency of approach in real, symbolic, missing value attributes makes it important. An instance based algorithm made for symbolic attributes fail in features of real value thus lacking in incorporated theoretical base. Approaches successful in feature of real values are thus in an ad-hoc fashion are made to handle symbolic attributes. Handling of missing values by classifiers poses a similar problem. Usually missing values treated as a separate value, thought as maximally different, substitute for average value, otherwise simply ignore them. Entropy based classifier is a solution for these issues (Painuli 2014).

3.7 Locally Weighted Learning (LWL)

The LWL uses an instance-based algorithm, assigns instance weights. This algorithm can perform both classification and regression (Jiawei and Kamber 2001). In this paper, it is used for a regression problem. The basic idea of the LWL is that any non-linearity can be approximated by a linear model, if the output surface is smooth. Therefore, instead of looking for a complex global model, it is easy to approximate non-linear functions by using simple local models (Arif *et al.* 2001).

3.8 DecisionStump

DecisionStump, constructs one-level binary decision trees for datasets with a categorical or numeric class, handling with missing values by treating them as a separate value and extending a third branch from the stump (Erdal and Karahanoglu 2016). It makes (1) regression based on mean-squared errors or (2) classification based on entropy depending on the data type to be predicted (Csépe *et al.* 2014). It also finds a single attribute that provides the best discrimination between the classes and then bases future predictions on this attribute (Iba 1992).

3.9 Model Trees Regression (M5P)

The M5P is a regression-based decision tree algorithm, which constructs a model tree using the M5 algorithm. For a given instance, the tree is traversed from top to bottom until a leaf node is reached. At each node in the tree, a decision is taken to follow a particular branch based on a test condition on the attribute associated with that node (Erdal 2013). Because each leaf nodes contains a linear regression model to obtain the predicted output, the tree is called a model tree (Wang and Witten 1997). An M5 tree is formed using a divide-and-conquer method. At each interior node, a test condition is selected that splits the instances into subsets based on the test outcome. An M5 tree can be complex. To simplify the tree without losing its basic functionality, it may be pruned. Starting from the bottom of the tree, the error is calculated for the linear model at each node. If the error is less than the model subtree owned by the node, then the subtree for this node is pruned. The major advantage of M5P over regression trees is that M5P are much smaller than regression trees, the decision strength is clear, and the regression functions do not normally involve many variables (Ekinci *et al.* 2011).

3.10 RandomForest

RandomForest built random forests by bagging ensembles of random trees (Erdal and Karahanoglu 2016). Namely, It build several individual classification trees using random samples of the data (i.e., bagging) and then vote for the most popular class (Breiman 2001). It is a hybrid of the Bagging algorithm and the random subspace method, and uses decision trees as the base classifier.

3.11 RandomTree

Table 4 Procedures considered estimating internal validity of predictive models

Evaluation Procedure	%	Method	Training sample	Test sample	Estimated performance	Repetitions
CV5	20%	k-fold cross validation	80% of original	Independent 20% of original	Test	5
CV10	10%	k-fold cross validation	90% of original	Independent 10% of original	Test	10
SS10	10%	Split sample validation	90% of original	Independent 10% of original	Average (test)	1
SS20	20%	Split sample validation	80% of original	Independent 20% of original	Average (test)	1

RandomTree is also known isa regression-based decision tree algo-rithm. Trees that are built by RTree consider Krandomly selected attributes at each node (without pruning). Furthermore, it has an option to allow estimation of class probabilities (or target mean inthe regression case) based on a hold-out set (backfitting) (Erdal and Karahanoglu 2016).

3.12 Reduced Error Pruning Tree (REPTree)

The REPTree algorithm generates a regression tree using the node statistics such as information gain or variance reduction measured in the top-down phase, and prunes it using reduced-error pruning (Portnoy and Koenker 1997). Optimized for speed, it only sorts values for numeric attributes once and it handles with missing values by splitting instances into small pieces. One can set the minimum number of instances per each leaf, maximum tree depth, minimum proportion of training set variance for a split (numeric classes only), and number of folds for pruning (Erdal and Karahanoglu 2016).

4. Results and discussion

4.1 Evaluation process

Four evaluation processes (10-fold cross validation, 5-fold cross validation, 10% split sample validation & 20% split sample validation) were used to examine the predictive models. A common way to measure the predictive performance on a test set is by means of a “split sample”, in which a subsample of the observation data is withheld from training and used to measure the accuracy of prediction. In *k*-fold cross-validation, the data are divided into *k* subsets of equal size. The regression technique is then applied *k* times, each time leaving out one of the subsets and using that subset to compute the prediction accuracy. Predictive performance is quantified by calculating model evaluation measures on the predicted values for cross-validation. Four internal validation procedures were evaluated estimate test performance more accurately (Table 4). Split-sample methods were referred to as the split-10% and split 20% methods, where 10% or 20% of the sample were kept as an independent evaluation part for the proposed regression models that were estimated on 90% or 80% of the sample, respectively. The split was made once and at random. Cross-validation was performed with 10% or 20% of the data consecutively serving as a test part. Models were estimated on one part of the data (90% or 80%, respectively) and tested on the independent part. The average performance was calculated over 10 or 5 repetitions, respectively.

4.2 Performance statistics

Twelve machine learning models proposed in this study were evaluated by using the five performance statistics (i.e., coefficient of determination (R^2), mean absolute error (MAE), root mean squared error (RMSE), relative absolute error (RAE) and root relative squared error (RRSE)) were calculated to investigate the statistical relation between original data and predicted data with highest preference for MAE, RMSE, RAE and RRSE closest to zero, whereas for R^2 closest to unity (Aydogmus *et al.* 2015, Aydogmus *et al.* 2015, Demirdogen *et al.* 2017).

Coefficient of determination (R^2)

$$R^2 = \left(\frac{n\sum y.y' - (\sum y)(\sum y')}{\sqrt{(\sum y^2) - (\sum y)^2}\sqrt{(\sum y'^2) - (\sum y')^2}} \right)^2 \tag{12}$$

where y=actual value; y'=predicted value; and n=number of data samples.

Root mean squared error (RMSE)

$$RMSE = \sqrt{\frac{\sum (y' - y)^2}{n}} \tag{13}$$

Mean absolute error (MAE):

$$MAE = \frac{1}{n} \sum_{i=1,n} |y - y'| \tag{14}$$

Relative absolute error (RAE)

$$RAE = \frac{\sum_{i=1}^{n} |y' - y|}{\sum_{i=1}^{n} |\bar{y} - y|} \tag{14}$$

Root relative squared error (RRSE)

$$RRSE = \sqrt{\frac{\sum_{i=1}^{n} (y' - y)^2}{\sum_{i=1}^{n} (\bar{y} - y)^2}} \tag{15}$$

4.3 Empirical results

The results of twelve machine learning models are summarized in Tables 5-9. A goodness-of-fit indicator as R^2 can be a good complement on the predictive performance measures and can provide important information for end-users more interested in showing the relationships present in the data, rather than developing good machine learning

Table 5 R^2 statics of predictive models

Machine learning	Model	R^2				descriptive statistics		
		CV5	CV10	SS10	SS20	mean	std dev	var
Functions	LR	0.9029	0.9065	0.8123	0.9019	0.8809	0.0458	0.0016
	SLR	0.8810	0.8783	0.7218	0.8896	0.8427	0.0807	0.0049
	MLP	0.9082	0.9021	0.8688	0.9088	0.8970	0.0190	0.0003
	SMOreg	0.9044	0.9054	0.8709	0.8974	0.8945	0.0162	0.0002
Lazy-learning algorithms	IBk	0.8957	0.8976	0.8482	0.8701	0.8779	0.0234	0.0004
	KStar	0.9069	0.9027	0.8030	0.8860	0.8747	0.0486	0.0018
	LWL	0.8823	0.8774	0.6249	0.6838	0.7671	0.1324	0.0132
Tree-based learning algorithms	DecisionStump	0.6244	0.6103	0.3588	0.6921	0.5714	0.1461	0.0160
	M5P	0.9029	0.9065	0.8879	0.9019	0.8998	0.0082	0.0000
	RandomForest	0.8987	0.8847	0.8310	0.8972	0.8779	0.0319	0.0008
	RandomTree	0.8530	0.8048	0.7903	0.8414	0.8224	0.0297	0.0007
	REPTree	0.8523	0.8543	0.7665	0.8352	0.8271	0.0413	0.0013

Table 6 MAE statics of predictive models

Machine learning	Model	MAE				descriptive statistics		
		CV5	CV10	SS10	SS20	mean	std dev	var
Functions	LR	0.5262	0.5083	0.6907	0.5219	0.5618	0.0863	0.0056
	SLR	0.5700	0.5767	0.8872	0.5651	0.6498	0.1584	0.0188
	MLP	0.5059	0.5124	0.5811	0.5152	0.5287	0.0352	0.0009
	SMOreg	0.5404	0.5313	1.1949	0.8415	0.7770	0.3137	0.0738
Lazy-learning algorithms	IBk	0.5312	0.5233	1.0840	0.7535	0.7230	0.2633	0.0520
	KStar	0.5451	0.5497	0.7702	0.6095	0.6186	0.1052	0.0083
	LWL	0.5906	0.5908	0.9906	0.8641	0.7590	0.2011	0.0303
Tree-based learning algorithms	DecisionStump	1.0136	1.0438	1.3805	0.9577	1.0989	0.1911	0.0274
	M5P	0.5262	0.5083	0.6242	0.5219	0.5452	0.0532	0.0021
	RandomForest	0.5599	0.5983	0.8744	0.5555	0.6470	0.1528	0.0175
	RandomTree	0.6379	0.7333	0.7617	0.7075	0.7101	0.0530	0.0021
	REPTree	0.6252	0.6461	0.8518	0.7152	0.7096	0.1023	0.0079

Table 7 RMSE statics of predictive models

Machine learning	Model	RMSE				descriptive statistics		
		CV5	CV10	SS10	SS20	mean	std dev	var
Functions	LR	0.6695	0.6571	0.9673	0.7075	0.7504	0.1462	0.0160
	SLR	0.7410	0.7496	1.1833	0.7342	0.8520	0.2209	0.0366
	MLP	0.6513	0.6723	0.7953	0.6808	0.6999	0.0648	0.0031
	SMOreg	0.6918	0.6830	1.4038	1.0455	0.9560	0.3430	0.0882
Lazy-learning algorithms	IBk	0.7073	0.6957	1.4056	1.0152	0.9560	0.3343	0.0838
	KStar	0.7051	0.7184	1.0362	0.8048	0.8161	0.1532	0.0176
	LWL	0.7368	0.7523	1.3422	1.2398	1.0178	0.3183	0.0760
Tree-based learning algorithms	DecisionStump	1.3192	1.3470	1.7794	1.2682	1.4285	0.2362	0.0419
	M5P	0.6695	0.6571	0.8210	0.7075	0.7138	0.0746	0.0042
	RandomForest	0.6846	0.7303	1.1454	0.7137	0.8185	0.2188	0.0359
	RandomTree	0.8354	0.9629	1.0189	0.9102	0.9319	0.0781	0.0046
	REPTree	0.8308	0.8208	1.0907	0.9230	0.9163	0.1250	0.0117

models. The R^2 static is summarized for each model in Table 5. In our study we attempt to simulate a naive user of machine learning who is not an expert in machine learning algorithms and does not know which parameters to tune to improve results. We believe that this is a more realistic scenario than those considered in other papers (and it also counteracts overfitting to the dataset at hand).

The performance of all eleven predictors except Decisionstump is relatively close, with the R^2 values ranging between 0.9088 and 0.8530.The best performance based on R^2, is obtained by the MLP and KStar, followed LR and M5P with same accuracy, respectively. REPTree, RandomTree and DecisionStump are scoring worse for this indicator. Clearly, the decision stump model yields the

Table 8 RAE statics of predictive models

Machine learning	Model	RAE (%)				descriptive statistics		
		CV5	CV10	SS10	SS20	mean	std dev	var
Functions	LR	28.20%	27.26%	36.60%	27.63%	29.92%	4.47%	0.15%
	SLR	30.54%	30.93%	47.02%	29.92%	34.60%	8.29%	0.52%
	MLP	27.11%	27.48%	30.79%	27.27%	28.16%	1.76%	0.02%
	SMOreg	28.96%	28.49%	63.32%	44.55%	41.33%	16.45%	2.03%
Lazy-learning algorithms	IBk	28.47%	28.06%	57.44%	39.89%	38.47%	13.79%	1.43%
	KStar	29.21%	29.48%	40.81%	32.27%	32.94%	5.43%	0.22%
	LWL	31.65%	31.67%	52.50%	45.74%	40.39%	10.45%	0.82%
Tree-based learning algorithms	DecisionStump	54.32%	55.98%	73.16%	50.70%	58.54%	9.99%	0.75%
	M5P	28.20%	27.30%	33.08%	27.63%	29.05%	2.71%	0.06%
	RandomForest	30.01%	32.09%	46.34%	29.41%	34.46%	8.00%	0.48%
	RandomTree	34.19%	39.33%	40.36%	37.45%	37.83%	2.71%	0.06%
	REPTree	33.51%	34.65%	45.14%	37.86%	37.79%	5.23%	0.21%

Table 9 RRSE statics of predictive models

Machine learning	Model	RRSE (%)				descriptive statistics		
		CV5	CV10	SS10	SS20	mean	std dev	var
Functions	LR	31.00%	30.39%	44.29%	31.21%	34.22%	6.72%	0.34%
	SLR	34.31%	34.67%	54.18%	32.38%	38.89%	10.24%	0.79%
	MLP	30.16%	31.10%	36.41%	30.03%	31.92%	3.03%	0.07%
	SMOreg	32.03%	31.59%	64.27%	46.11%	43.50%	15.40%	1.78%
Lazy-learning algorithms	IBk	32.75%	32.18%	64.36%	44.77%	43.51%	15.06%	1.70%
	KStar	32.65%	33.23%	47.44%	35.50%	37.20%	6.94%	0.36%
	LWL	34.12%	34.79%	61.45%	54.68%	46.26%	13.91%	1.45%
Tree-based learning algorithms	DecisionStump	61.08%	62.30%	81.47%	55.93%	65.20%	11.19%	0.94%
	M5P	31.00%	30.39%	37.59%	31.21%	32.55%	3.38%	0.09%
	RandomForest	31.70%	33.78%	52.44%	31.48%	37.35%	10.12%	0.77%
	RandomTree	38.68%	44.54%	46.65%	40.14%	42.50%	3.72%	0.10%
	REPTree	38.47%	37.96%	49.94%	40.71%	41.77%	5.57%	0.23%

Table 10 Best evaluation processes

Machine learning	Model	R^2	MAE	RMSE	RAE	RRSE
Functions	LR	CV10	CV10	CV10	CV10	CV10
	SLR	SS20	SS20	SS20	SS20	SS20
	MLP	SS20	CV5	CV5	CV5	SS20
	SMOreg	CV10	CV10	CV10	CV10	CV10
Lazy-learning algorithms	IBk	CV10	CV10	CV10	CV10	CV10
	KStar	CV5	CV5	CV5	CV5	CV5
	LWL	CV5	CV5	CV5	CV5	CV5
Tree-based learning algorithms	DecisionStump	SS20	SS20	SS20	SS20	SS20
	M5P	CV10	CV10	CV10	CV10	CV10
	RandomForest	CV5	SS20	CV5	SS20	SS20
	RandomTree	CV5	CV5	CV5	CV5	CV5
	REPTree	CV10	CV5	CV10	CV5	CV10

worst accuracy. This is expected because it only uses one attribute in the data for prediction.As measures for accuracy, MAE, RMSE, RAE and RRSE are calculated of every regression methods.

To check the validity of the comparisons between the regression models, MAE, RMSE, RAE and RRSE were also calculated. The obtained results were almost identical to R^2 results. Tables 6-9, demonstrated that MLP, LR and M5P produce the lowest MAE, RMSE, RAE and RRSE, respectively.

Four evaluation processes (10-fold cross validation, 5-fold cross validation, 10% split sample validation & 20% split sample validation) were used to evaluate the predictive models. Table 10, summarized the best evaluation process according to performance statics. The best results of LR, SLR, SMOreg, IBk, KStar, LWL, DecisionStump, M5P and RandomTree are obtained by using same evaluation process, individually. On the other hand, MLP, RandomForest and REPTree yields the best results of performance statics using different evaluation processes.

The accuracy of the proposed machine learning models is assessed graphically with box-plots. Box plots are useful for identifying outliers and for comparing distributions. In a prediction problem, the distribution of the actual and predicted values demonstates the prediction performance of the utilized methods. So, the actual and predicted strength distributions of the models are depicted with box plots presented in Figs. 4-7. The boxes indicate the interquartile ranges (5th and 95th percentile of actual and predicted data), dots indicate values outside the range and the horizontal line within each boxes indicate the median values.

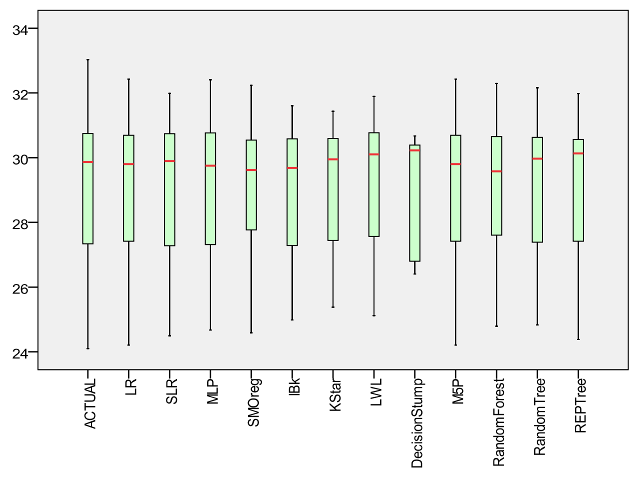

Fig. 4 Box plots of actual strength and predicted strength distributions of the predictive models for 5-fold cross validation evaluation process

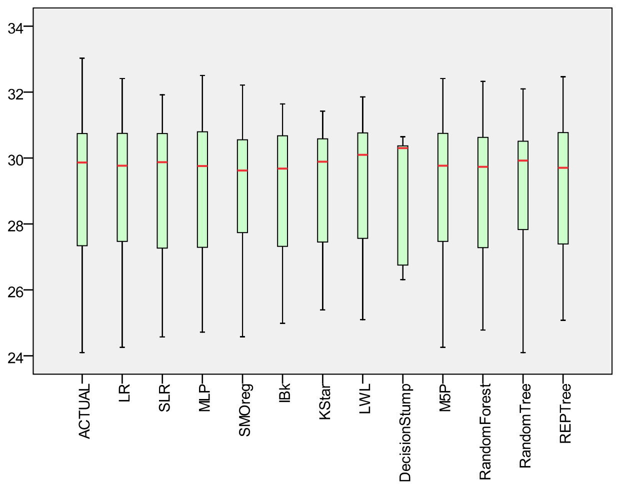

Fig. 5 Box plots of actual strength and predicted strength distributions of the predictive models for 10-fold cross validation evaluation process

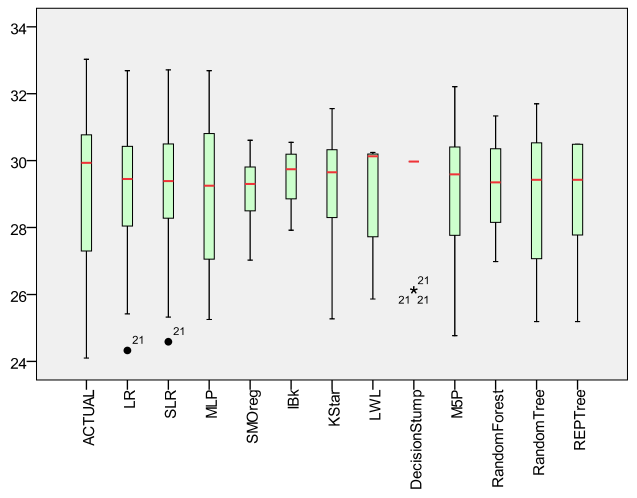

Fig. 6 Box plots of actual strength and predicted strength distributions of the predictive models for 10% split sample validation evaluation process

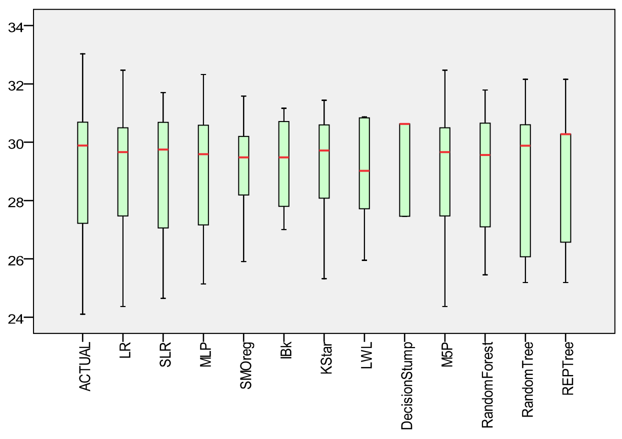

Fig. 7 Box plots of actual strength and predicted strength distributions of the predictive models for 20% split sample validation evaluation process

5. Conclusions

In literature, there are many previously suggested single and multi variable equations used for the prediction of compressive strength of concrete utilizing nondestructive test results.

In this paper, machine learning techniques have been explored to develop models that predict the compressive strength of concrete using experimental data.

To the best our knowledge, it is the first so comprehensive comparative study that handles the concrete compressive strength prediction problem via twelve different algorithms.

In this study, twelve artificial intelligence (AI) models compared for predicting the concrete compressive strength. These models can be divided into three categories (i) Functions (i.e., Linear Regression, Simple Linear Regression, Multilayer Perceptron, Support Vector Regression), (ii) Lazy-Learning Algorithms (i.e., IBk Linear NN Search, KStar, Locally Weighted Learning) (iii) Tree-Based Learning Algorithms (i.e., Decision Stump, Model Trees Regression, Random Forest, Random Tree, Reduced Error Pruning Tree).

Four evaluation processes 4 validation implements (i.e., 10-fold cross validation, 5-fold cross validation, 10% split sample validation & 20% split sample validation) are used to examine the predictive models.

The study shows that the methods are quite successful in the prediction of the compressive strength of concrete and the predicted values are very close to the real measurements.

Clearly, the multilayer perceptron model was found to be the best and the decision stump technique was found to be the poorest performing predictive model.

It's obtained that prediction success of multilayer perceptron has been found more satisfactory than the other's. It is concluded that the multilayer perceptron can be used effectively as an alternative method by researchers and the investors for predicting the compressive strength of concrete.

While there are many pros and cons of each approach, here we only deal with the accuracy aspect through an experimental study. The other considerations, though very important, are outside the scope of this research.

References

Altun, F., Kisi, O. and Aydin, K. (2008), "Predicting the compressive strength of steel fiber added lightweight concrete using neural network", *Comput. Mater. Sci.*, **42**(2), 259-265.

Antonaci, P., Bruno, C.L.E., Gliozzi, A.S. and Scalerandi, M. (2010), "Monitoring evolution of compressive damage in concrete with linear and nonlinear ultrasonic methods", *Cement Concrete Res.*, **40**, 1106-1113.

Arif, M., Ishihara, T. and Inooka, H. (2001), "Incorporation of experience in iterative learning controllers using locally weighted learning", *Autom*, **37**, 881-888.

ASTM C39 (2001), Standard Test Method for Compressive Strength of Cylindrical Concrete Specimens, Annual Book of ASTM Standards, Philadelphia, USA.

ASTM C42 (1999), Standard Test Method for Obtaining and Testing Drilled Cores and Sawed Beams of Concrete, Annual Book of ASTM Standards, Philadelphia, USA.

ASTM C597 (1998), Standard Test Method for Pulse Velocity Through Concrete, Annual Book of ASTM Standards, Philadelphia, USA.

ASTM C803 (1999), Standard Test Method for Penetration Resistance of Hardened Concrete, Annual Book of ASTM Standards, Philadelphia, USA.

ASTM C805 (1997), Standard Test Method for Rebound Number of Hardened Concrete, Annual Book of ASTM Standards, Philadelphia, USA.

Atici, U. (2011), "Prediction of the strength of mineral admixture concrete using multivariable regression analysis and an artificial neural network", *Exp. Syst. Appl.*, **38**, 9609-9618.

Aydin, F. and Saribiyik, M. (2010), "Correlation between Schmidt hammer and destructive compressions testing for concretes in existing buildings", *Sci. Res. Essay.*, **5**, 1644-1648.

Aydogmus, H.Y., Ekinci, A., Erdal, H.İ. and Erdal, H. (2015), "Optimizing the monthly crude oil price forecasting accuracy via bagging ensemble models", *J. Econ. Int. Finance*, **7**(5), 127-136.

Aydogmus, H.Y., Erdal, H.İ., Karakurt, O., Namli, E., Turkan, Y.S. and Erdal, H. (2015), "A comparative assessment of bagging ensemble models for modeling concrete slump flow", *Comput. Concrete*, **16**(5), 741-757.

Baykan, U.N., Erdal, M. and Ugur, L.O. (2017), "A fuzzy logic model for prediction of compressive strength of concrete by use of non-destructive test results", *Rev. Rom. Mater.*, **47**(1), 54-59.

Betrie, G.D., Tesfamariam, S., Morin, K.A. and Sadiq, R. (2013), "Predicting copper concentrations in acid mine drainage: A comparative analysis of five machine learning techniques", *Environ. Monit. Assess.*, **185**(5), 4171-4182.

Breiman, L. (2001), "Random forests", *Mach. Learn.*, **45**(1), 25-32.

Breysse, D. (2012), "Nondestructive evaluation of concrete strength: An historical review and a new perspective by combining NDT methods", *Constr. Build. Mater.*, **33**, 139-163.

Cheng, M.Y. and Wu, Y.W. (2009), "Evolutionary support vector machine inference system for construction management", *Automat. Constr.*, **18**, 597-604.

Cheng, M.Y., Chou, J.S., Roy, A.F.V. and Wu, Y.W. (2012), "High-performance concrete compressive strength prediction using time-weighted evolutionary fuzzy support vector machines inference model", *Automat. Constr.*, **28**, 106-115.

Cleary, J.G. and Trigg, L.E. (1995), "K*: an instance-based learner using an entropic distance measure", *Proceedings of the 12th International Conference on Machine Learning*, 108-114.

Csépe, Z., Makra, L., Voukantsis, D., Matyasovszky, I., Tusnády, G., Karatzas, K. and Thibaudon, M. (2014), "Predicting daily ragweed pollen concentrations using computational intelligence techniques over two heavily polluted areas in Europe", *Sci. Total Environ.*, **476**, 542-52.

Demirdogen, O., Erdal, H. and Akbaba, A.İ. (2017), "Comparing various machine learning methods for prediction of patient revisit intention: A case study", *SUJEST*, **5**(4), 386-401.

Dias, W. and Pooliyadda, S. (2001), "Neural networks for predicting properties of concretes with admixtures", *Constr. Build. Mater.*, **15**(7), 371-379.

Domone, P. and Soutsos, M. (1994), "An approach to the proportioning of high-strength concrete mixes", *Concrete Int.*, **16**, 26-31.

Ekinci, S., Celebi, U.B., Bala, M., Amasyali, M.F. and Boyaci, U.K. (2011), "Predictions of oil/chemical tanker main design parameters using computational intelligence techniques", *Appl. Soft. Comput.*, **11**, 2356–2366.

Erdal, H. (2015), "Makine öğrenmesi yöntemlerinin inşaat sektörüne katkısı: Basınç dayanımı tahminlemesi", *Pamukkale Univ Muh Bilim Derg*, **21**(3), 109-114.

Erdal, H. and Karahanoglu, İ. (2016), "Bagging ensemble models for bank profitability: An emprical research on Turkish development and investment banks", *Appl. Soft. Comput.*, **49**, 861-867.

Erdal, H.İ. (2013), "Two-level and hybrid ensembles of decision trees for high performance concrete compressive strength prediction", *Eng. Appl. Artif. Intell.*, **26**(7), 1689-1697.

Erdal, H.İ. and Karakurt, O. (2013), "Advancing monthly stream flow prediction accuracy of CART models using ensemble learning paradigms", *J. Hydrol.*, **477**, 119-128.

Erdal, H.İ., Karakurt, O. and Namlı, E. (2013), "High performance concrete compressive strength forecasting using ensemble models based on discrete wavelet transform", *Eng. Appl. Artif. Intell.*, **26**(4), 1246-1254.

Erdal, M. (2002), "Determination of compressive strength of concrete by some non-destructive test methods", M.Sc. Thesis, Gazi University, Ankara.

Erdal, M. (2009), "Prediction of the compressive strength of vacuum processed concretes using artificial neural network and regression techniques", *Sci. Res. Essay.*, **4**(10), 1057-1065.

Erdal, M. and Simsek, O. (2006), "Investigation of the performance of some non-destructive tests on the determination of compressive strength of vacuum-processed concrete", *J. Fac. Eng. Arch. Gazi Univ.*, **21**(1), 65-73.

Galan, A. (1967), "Estimate of concrete strength by ultrasonic pulse velocity and damping constant", *ACI J. Proc.*, **64**(10), 678-684.

Gupta, R., Kewalramani, M.A. and Goel, A. (2006), "Prediction of concrete strength using neural-expert system", *J. Mater. Civil Eng.*, **18**(3), 462-466.

Haykin, S. (1999), *Neural Networks, A Comprehensive Foundation*, 2nd Edition, Prentice Hall.

Hola, J. and Schabowicz, K. (2005), "Application of artificial neural networks to determine concrete compressive strength based on non-destructive tests", *J. Civil Eng. Manage.*, **11**, 23-32.

Iba W.L. (1992), "Induction of one-level decision trees", *Ninth International Conference on Machine Learning*, Aberdeen.

Jiawei, H. and Kamber, M. (2001), *Data Mining: Concepts and Techniques*, Morgan Kaufmann.

Kewalramani, M.A. and Gupta, R. (2006), "Concrete compressive strength prediction using ultrasonic pulse velocity through artificial neural networks", *Automat. Constr.*, **15**, 374-379.

Kheder, G.F. (1998), "A two stage procedure for assessment of in-situ concrete strength using combined non-destructive testing", *Mater. Struct.*, **32**, 410-417.

Lee, S.C. (2003), "Prediction of concrete strength using artificial neural networks", *Eng. Struct.*, **25**(7), 849-857.

Mehta, P.K. and Monterio, P.J.M. (2006), *Concrete Structure, Properties and Materials*, 3th Edition, Mc Graw-Hill Companies.

Mielentz, F. (2008), "Phased arrays for ultrasonic investigations in concrete components", *J. Nondestruct. Eval.*, **27**, 23-33.

Mousavi, S.M., Gandomi, A.H., Alavi, A.H. and Vesalimahmood, M. (2010), "Modeling of compressive strength of HPC mixes using a combined algorithm of genetic programming and orthogonal least squares", *Struct. Eng. Mech.*, **36**, 225-241.

Namlı, E., Erdal, H.İ. and Erdal, H. (2016), "Dalgacık dönüşümü ile beton basınç dayanım tahmininin iyileştirilmesi", *Politeknik Dergisi*, **19**(4), 471-480.

Neville, A.M. (1993), *Properties of Concrete*, 3th Edition, Longman Scientific & Technical.

Painuli, S., Elangovan, M. and Sugumaran, V. (2014), "Tool condition monitoring using K-star algorithm", *Exp. Syst. Appl.*, **41**, 2638-2643.

Portnoy, S. and Koenker, R. (1997), "The Gaussian hare and the Laplacian tortoise: computability of squared-error versus absolute-error estimators", *Stat. Sci.*, **12**(4), 279-300.

Qasrawi, H.Y. (2000), "Concrete strength by combined nondestructive methods simply and reliable predicted", *Cement Concrete Res.*, **30**, 739-746.

Rajasekaran, S. and Amalraj, R. (2002), "Predictions of design parameters in civil engineering problems using SLNN with a single hidden RBF neuron", *Comput. Struct.*, **80**, 2495-2505.

Rajasekaran, S. and Lavanya, S. (2007), "Hybridization of genetic algorithm with immune system for optimization problems in structural engineering", *Struct. Multidisc. Optim.*, **34**, 415-429.

Rajasekaran, S., Suresh, D, and Pai, GAV. (2002), "Application of sequential learning neural networks to civil engineering modeling problems", *Eng. Comput.*, **18**, 138-147.

Ramyar, K. and Kol, P. (1996), "Destructive and non-destructive test methods for estimating the strength of concrete", *Cement Concrete World*, **2**, 46-54.

Saridemir, M., Topcu, I.B., Ozcan, F. and Severcan, M.H. (2009) "Prediction of longterm effects of GGBFS on compressive strength of concrete by artificial neural networks and fuzzy logic", *Constr. Build. Mater.*, **23**(3), 1279-1286.

Sobhani, J., Najimi, M., Pourkhorshidi, A.R. and Parhizkar, T. (2010), "Prediction of the compressive strength of no-slump concrete: A comparative study of regression, neural network and ANFIS models", *Constr. Build. Mater.*, **24**(5), 709-718.

Solis-Carcano, R. and Moreno, EI. (2008), "Evaluation of concrete made with crushed limestone aggregate based on ultrasonic pulse velocity", *Constr. Build. Mater.*, **22**, 1225-1231.

Subasi, S. (2009), "Prediction of mechanical properties of cement containing class C fly ash by using artificial neural network and regression technique", *Sci. Res. Essay.*, **4**(4), 289-297.

Topcu, I.B. and Saridemir, M. (2008), "Prediction of compressive strength of concrete containing fly ash using artificial neural network and fuzzy logic", *Comput. Mater. Sci.*, **41**(3), 305-311.

Trtnik, G., Kavcic, F. and Turk, G. (2009), "Prediction of concrete strength using ultrasonic pulse velocity and artificial neural networks", *Ultrasonic.*, **49**, 53-60.

Tufekci, P. (2014), "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning method", *Int. J. Elect. Power Energy Syst.*, **60**, 126-140.

Turkan, Y.S., Aydogmus, H.Y. and Erdal, H. (2016), "The prediction of the wind speed at different heights by machine learning methods", *IJOCTA*, **6**(2), 179-187.

Wang, W. and Xu, Z. (2004), "A heuristic training for support vector regression", *Neurocomput.*, **61**, 259-275.

Wang, Y. and Witten, I. (1997), "Inducing model trees for continuous classes", *Ninth European Conference on Machine Learning*, Prague, Czech Republic.

Windsor Probe Test System Inc. (1994), WPS 500 Windsor Probe Test System Operating Instructions.

Witten, I.H. and Frank, E. (2005), *Data Mining: Practical Machine Learning Tools And Technique*, Morgan Kaufmann Publishers.

Yaprakli, T.S. and Erdal, H. (2015), "Bankacılık sektöründe pazarlama karması elemanlarının önceliklerinin belirlenmesi: Erzurum ili örneği", *JASSS*, **38**, 481-500.

Yeh, I.C. (1998), "Modeling of strength of high-performance concrete using artificial neural networks", *Cement Concrete Res.*, **28**, 1797-1808.

Yeh, I.C. (2007), "Modeling slump flow of concrete using second-order regressions and artificial neural networks", *Cement Concrete Compos.*, **29**, 474-480.

Yeh, I.C. and Lien, L.C. (2009), "Knowledge discovery of concrete material using genetic operation trees", *Exp. Syst. Appl.*, **36**(3), 5807-5812.

Zarandi, M.F., Türksen, I.B., Sobhani, J. and Ramezanianpour, A.A. (2008), "Fuzzy polynomial neural networks for approximation of the compressive strength of concrete", *Appl. Soft. Comput.*, **8**(1), 488-498.

CC