Journal of Soft Computing in Civil Engineering 5-4 (2021) 90-113

Contents lists available at SCCE

Journal of Soft Computing in Civil Engineering

Journal homepage: www.jsoftcivil.com

Multiple Target Machine Learning Prediction of Capacity Curves of Reinforced Concrete Shear Walls

Yicheng Yang[1], In Ho Cho[2*]

1. Ph.D. Candidate, Civil, Construction and Environmental Engineering, Iowa State University, Iowa, United States
2. Associate Professor, Civil, Construction and Environmental Engineering, Iowa State University, Iowa, United States

Corresponding author: *icho@iastate.edu*

https://doi.org/10.22115/SCCE.2021.314998.1381

ARTICLE INFO

Article history:
Received: 13 November 2021
Revised: 22 November 2021
Accepted: 22 November 2021

Keywords:
Machine learning for capacity curve prediction;
Multiple-target regression model;
Clus;
Shear wall database;
Uncertainty quantification.

ABSTRACT

Reinforced concrete (RC) shear wall is one of the most widely adopted earthquake-resisting structural elements. Accurate prediction of capacity curves of RC shear walls has been of significant importance since it can convey important information about progressive damage states, the degree of energy absorption, and the maximum strength. Decades-long experimental efforts of the research community established a systematic database of capacity curves, but it is still in its infancy to productively utilize the accumulated data. In the hope of adding a new dimension to earthquake engineering, this study provides a machine learning (ML) approach to predict capacity curves of the RC shear wall based on a multi-target prediction model and fundamental statistics. This paper harnesses bootstrapping for uncertainty quantification and affirms the robustness of the proposed method against erroneous data. Results and validations using more than 200 rectangular RC shear walls show a promising performance and suggest future research directions toward data- and ML-driven earthquake engineering.

How to cite this article: Yang Y, Cho IH. Multiple target machine learning prediction of capacity curves of reinforced concrete shear walls. J Soft Comput Civ Eng 2021;5(4):90–113. https://doi.org/10.22115/scce.2021.314998.1381

1. Introduction

In the past decades, persistent efforts have been devoted to gaining insights into the nonlinear behaviors of damaged rectangular RC walls [1–4]. Driven by these accomplishments, the research community benefits from databases (e.g., ACI 445B Shear Wall Database, Peer Structural Performance Databases, and DesignSafe Platform). Many ML-based predictions of RC structures have been on trial [5,6]. ML gives computers the ability to learn complex data without explicitly programmed rules. ML can be categorized into single-target prediction and multiple-target prediction methods regarding the number of prediction targets. There exist various applications of single-target ML methods in infrastructure engineering. The prediction of the shear strength of a deep beam was conducted by support vector regression (SVR). The researchers modified the SVR algorithm to optimize hyperparameters to be more suitable for civil applications [7,8]. Valdebenito [9] estimated the in-plane shear strength of reinforced masonry (RM) using the artificial neural network (ANN). ANN model was trained and tested by 285 RM walls from pieces of literature. The compressive strength of high-performance concrete had been predicted using the ensemble method [10]. Furthermore, with the interest of vertical structural elements, prediction of horizontal forces was made via support vector machine and ANN [5,11].

However, ML-based prediction of force-displacement (F-D) capacity curves is challenging since it involves multiple-target predictions. Two rare examples of curve prediction include predictions of soil-water characteristic curves (SWCCs) using genetic programming (GP) [12] and ANN [13], respectively. In Johari's work, SWCC itself was learned and predicted by the GP, but the final prediction is a complex mathematical expression of the curve. Sajib developed ANN models of the SWCC fitting parameters to predict the suction-water content relationship.

In this paper, we adopt a multi-target regression model (MTRM) to predict the capacity curves of RC shear walls. This paper is structured as follows. The second section demonstrates the methodology of the MTRM and its extension with ensemble learning. The third section presents complete procedures to build the capacity curve database and perform capacity curve prediction. The fourth section summarizes predictive results, validation, and impact of the extended database and erroneous data on the proposed method. Finally, the last section yields the conclusion and discusses the limitations and future extensions.

2. Multi-target regression model

MTRM has been implemented in the open-source machine learning system (named Clus) developed by Struyf [14]. Clus is a decision tree learner and rule learning system that works in the predictive clustering trees (PCTs) [14]. Prior to the demonstration of MTRM, it is instructive to introduce the background of ML. There are two categories of ML methods depending on training data. The first category is "supervised" learning, in which ML trains with data consisting of a pair of $\{\boldsymbol{x}^{(l)}, \boldsymbol{y}^{(l)}\}$ that stands for a vector of descriptive variables and $\boldsymbol{y}^{(l)} \in \mathbb{R}^k$ represents a target vector. The superscript (l) indicates labeled data. Contrarily, "unsupervised" learning trains ML with unlabeled data consisting of $\{\boldsymbol{x}^{(u)}\}$ where (u) indicates unlabeled data. A

decision tree, a typical supervised learning method, is a tree-shaped graph that uses a branching method to demonstrate every possible outcome of a decision. It is widely used in data mining to simplify complex problems. It usually starts with a single node, which branches into all possible outcomes.

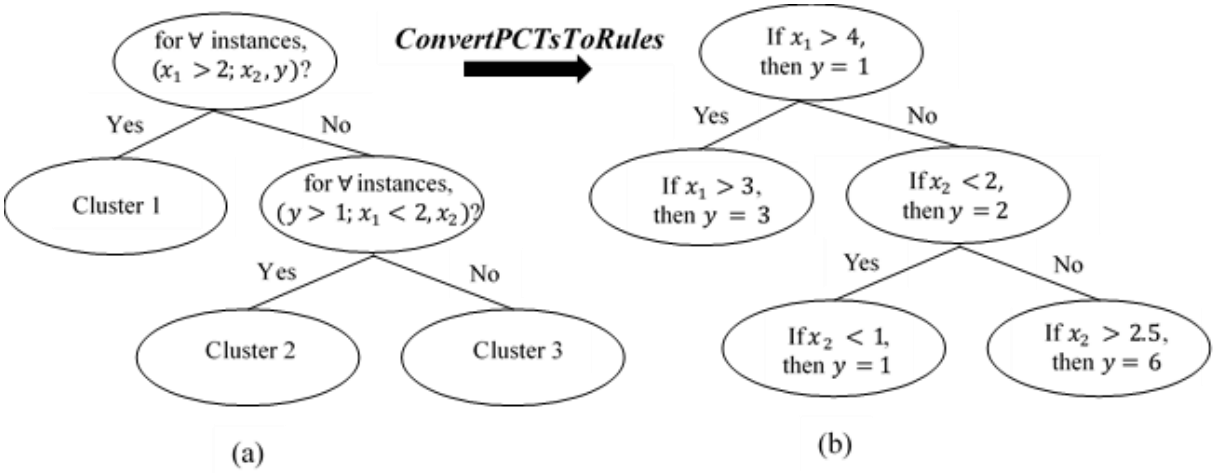


Fig. 1. (a) Illustrative example of a PCT; (b) example of a rule ensemble.

Each of those outcomes will branch into other nodes, which represent other possibilities. Clustering, a representative unsupervised learning method, tries to find a collection of points that are similar to each other in terms of homogeneous values of all variables compared with points out of the cluster. Decision tree and clustering are therefore considered as quite different methods. Decision trees partition instances to subsets in terms of values of target attributes only, and clustering splits instances to subclusters regarding the value of all descriptive attributes. Noteworthy, a PCT is a decision tree whose leaves do not contain classes, and each node, as well as each leaf, corresponds to a cluster in Fig. 1 (a) with instances in the form of $\{x_1, x_2, y\}$. Diversely, PCTs search for subsets with the values of both descriptive attributes and target attributes [15]. MTRM shares the same algorithm with PCTs in the context of constructing clusters. PCTs can be built with a standard "top-down induction of decision trees" (TDIDT) algorithm [16]. Top-down PCTs shape in a triangle whose root is up. All instances locate at the root at the beginning, and they are partitioned into subclusters by tests. The pseudo algorithm of constructing PCTs is presented in Table 1 [17].

It is instructive to recap key strategies of PCTs, i.e., a splitting criterion, a stopping criterion, and a pruning strategy, respectively. There are many splitting criteria (e.g., Shannon entropy [18] and Gain Ratio [19]). The purpose of splitting clustering is to obtain subclusters such that intra-cluster distance (the distance between examples belonging to different clusters) is minimized. For regression problems, intra-cluster distance is specified as the intra-cluster variance. Given a cluster and a test that will result in a partition of the cluster to decrease the variance, the intra-cluster variance is defined as:

$$var = \sum_{i=1}^{N} d(\boldsymbol{x}_i, \overline{\boldsymbol{x}})^2 \tag{1}$$

where $\overline{\boldsymbol{x}} \in \mathbb{R}^n$ is the mean vector of the cluster, and $\boldsymbol{x}_i \in \mathbb{R}^n$ $(i = 1, \cdots, N)$ is an element in the cluster, and N is the total number of elements in the cluster. The entity d stands for the Euclidean distance. Growing trees without stopping criteria will lead to an overfitting problem. Often, a test is applied to check whether the class distribution in the sub-clusters differs significantly. Since the regression problem uses intra-cluster variance as the heuristic for choosing the best split, then a reasonable stopping criterion is to use an F-test to check whether variance decreased significantly, and thus a test will be found.

Table 1
Algorithm of constructing PCTs.

1: Function ***PCT***(Training instances I):	9: Function ***BT***(I):
2: $(t^*, p^*) = \boldsymbol{BT}\ (I)$;	10: $p =$ partition induced on I by t ;
3: If $t^* \neq$ none	11: $(t^*, p^*, h^*) = (\text{None}, 0.5, 0)$;
4: for each $I_k \in P^*$	12: $h = \boldsymbol{var}(I) - \sum_{I_k \in p} \frac{\lvert I_k \rvert}{\lvert I \rvert} \boldsymbol{var}(I_k)$;
5: $Tree_k = PCT(I_k)$;	13: for each test
6: return node$(t_k, Tree_k)$;	14: if $(h > h^*)$
7: else if	15: $(t^*, p^*, h^*) = (t, p, h)$;
8: return leaf $(I_{protoype})$;	16: return (t^*, p^*) ;

If no acceptable test is found, the algorithm labels the leaf with the prototype instances and stops the growth. Pruning strategy is a technique to remove trivial parts of the tree to identify instances. Often pruning is done randomly for large data. This paper does not adopt any pruning strategies due to our small database size. The illustration of the pseudo-algorithm of constructing PCTs will help engineers with a comprehensive understanding of the MTRM. The ***PCT*** function takes instances I as input to grow trees. An instance represents a row of the dataset in this paper. The function ***PCT*** in line 1 of Table 1 is the algorithm's main function, which grows the decision tree until stopping criteria are met. The function ***BT*** is invoked in line 2 to search for the best test to partition training instances to hierarchical clusters. ***BT*** returns optimal t and p, denoted as (t^*, p^*), where t is an action test of attribute values to induce a partition on I, p is a partition induced on I by t (e.g., In Fig. 1 (a), a test t on root node checks whether x_1 is larger than two or not to partition I at the root to two sub-clusters via a partition p). The superscript "*" represents the optimal (i.e., best-so-far) quantities. With ***BT*** in line 2, ***PCT*** function is invoked recursively to obtain trees and the corresponding nodes within the loop in lines 5 and 6. However, if the best test is not found in line 7, then the algorithm will return a leaf labeled as the prototype instances in line 8. Usually, the prototype instances have the lowest average distance to all other instances in the cluster, such as the mean of the original instances.

Function ***BT*** is explained in the right column of Table 1. ***BT*** searches for the best test to partition the cluster to minimize intra-cluster variance (i.e., maximize inter-cluster variance). In line 11, the candidates for the best test (t^*) along with the corresponding partition (p^*) and heuristic value (h^*) are initialized. Here, h is defined in line 12, meaning a heuristic value of t. Function

var is defined in Eq. (1). Since t^* is initially unknown, h^* is set as zero. The loop in line 13 calculates heuristic values of all possible tests to partition clusters. The best test and partition will be chosen if a current heuristic value h is larger than the initial heuristic value h^* (lines 14-15).

2.1. Ensemble method

An ensemble method has been used to boost the prediction accuracy of this study. This method generates an ensemble of prediction models since combining a number of predictions is often more accurate than an individual prediction model [20,21].

Table 2
Pseudo code of ensemble method.

1: Let $\mathbf{M}$ = the original training data; n_{pm} = number of prediction models; $\mathbf{X}$ = the test data
2: for $i = 1\ to\ n_{pm}$ do
3: Create an identical training set $\mathbf{M_i}$ from $\mathbf{M}$
4: Build a prediction model PM_i with $\mathbf{M_i}$
5: end
6: for each test record $x_j \in \mathbf{X},\ j = 1, \dots, n$ do
7: $PM_{final}(x_j) = \frac{\sum_{i=1}^{n_{pm}} PM_i(x_j)}{n_{pm}}$
8: end

The general procedures for the ensemble method are summarized in Table 2. In line 3 of Table 2, the main loop creates n_{pm} sets of training data $\mathbf{M_1}$, …, $\mathbf{M_{n_{pm}}}$ by the simple random sampling method. It is a naive sampling method that generates every possible sample $\mathbf{M_i}$ of size $\frac{M}{n_{pm}}$ from the population of size M [22]. Each instance has an equal probability of being selected. Line 4 utilizes sets of training data to train n_{pm} base prediction models PM_1, …, $PM_{n_{pm}}$. Then line 7 aggregates predictions of all the models and algebraically averages these predictions as the final output for the regression problem. Various approaches have been successfully applied to construct ensemble learning. The popular ones are bootstrap aggregation (so-called bagging), boosting, and random forests. Bagging, a technique to generate multiple repeated bootstrap samples with replacement, is frequently used in classification and regression to improve stability and accuracy [23]. Instead of generating a succession of independent bootstrap samples, boosting trains multiple base prediction models using a weighted data set. Weights of samples are adjusted by issuing more weights on misclassified samples [24]. In this paper, random forests are implemented according to the research conclusion by Dragi, which indicates that multi-objective random forests are significantly better than multi-objective bagging [25]. Random forests share the same general procedures with other ensemble methods in Table 2. The general procedures to build random forests are shown as follows:

1. Subsets training data $\mathbf{M}$ to i bootstrap samples $\mathbf{M}_1, \dots, \mathbf{M}_i$ in line 3 of Table 2.

2. Build i decision trees $DT_1, \dots, DT_i$ with corresponding $\mathbf{M}_i$ as suggested in line 4. At each node, variables are selected at random out of all the features, and the best splits on these variables are used to split the node. Each tree is growing to the largest extent without pruning.

3. Perform prediction with test data using each tree DT_i in line 7. The final prediction will be the average of $PM_1(x_i)$, $PM_2(x_i)$, ... $PM_n(x_i)$ because it is a regression problem ($PM_i(x_i)$ is the prediction from decision tree DT_i).

In this paper, random forests have been employed as an ensemble learning method to cooperate with MTRM. Random forest cooperates with MTRM mainly in terms of two aspects. Firstly MTRM generates a collection of PCTs by bagging random forests instead of a single decision tree. Secondly, MTRM randomly picks attributes as input for function ***BT*** in Table 1 instead of using all attributes to find out the best test to partition the cluster.

Table 3
Algorithm of constructing rule ensembles. Note that *I* is training instances, and *T* is a collection of PCTs. *R* and *W* represent the collection of rules generated from *T* and their corresponding weights.

1: ***GenerateSetOfPCTs***(I):	5: ***OptimizeWeights***(R, I):
2: return T;	6: If (weight of $r \in R = 0$)
3: ***ConvertPCTsToRules***(T):	7: remove r;
4: return R;	8: return (R, W);

2.2. Rule ensemble for MTRM

Large ensembles of PCTs are hard to interpret. Thus, all PCTs are transcribed into a collection of rules. Rule learning, a collection of unordered rules whose predictions are combined via weighted voting, is an expressive and human-readable model representation. It is a conjunction of statements along with input variables. To briefly explain how the rule ensemble interprets the MTRM, the key algorithm to achieve rule ensembles of MTRM is summarized in Table 3 [26]. In line 1 of Table 3, function ***GenerateSetOfPCTs*** recursively calls function ***PCT*** in Table 1 to generate bagging of PCTs, then line 2 returns a collection of PCTs. Such large ensembles of PCTs are impossible to interpret, and thus all the trees are transcribed to sets of rules by function ***ConvertPCTsToRules*** in line 3 [27]. Line 5 finds the optimized weight for each of those rules *R* by function ***OptimizeWeights***. During this process, it is trying to assign as many weights as possible to zero, in the purpose of learning small and interpretable trees. A gradient-directed optimization algorithm [26] optimizes all the weights. The physical meaning of weights indicates the importance of each rule contributing to the final prediction. Lines 6 and 7 remove the trees if their optimal weights are zero. Finally, line 8 returns a collection of rules whose weights are not zero and their corresponding weights. Hence, the final prediction can be computed by the following equation:

$$\hat{y} = w_0 avg + \sum_{i=1}^{M} w_i r_i(x) \tag{2}$$

where w_0 is the baseline prediction, part (avg) is a constant vector with the average over all the targets. The entity r_i is a vector function which gives out a constant prediction shown in Fig. 1 (b) as a toy example. And w_i is the corresponding weight of a rule. Note that M indicates the number of rules in a PCT. Fig. 1 (a) considers a population of instances with two descriptive variables in the form of $\{x_1, x_2\}$ and a target response $\{y\}$. A toy PCT is constructed on top of

founded tests, and each clustering of the PCT is represented by a conditional statement as a result of function ***ConvertPCTsToRules*** in Fig. 1 (b). A prediction of y with $\{x_1, x_2\}$ = {5, 0.1}is calculated as:

$$\hat{y} = 0.95(1) + 0.2\ [\text{if}(x_1 > 4),\ \text{then}\ (1)] + 0.4\ [\text{if}(x_1 > 3),\ \text{then}\ (3)] + 0\ [\text{if}(x_2 < 2),\ \text{then}\ (2)] + 0.3\ [\text{if}(x_2 < 1),\ \text{then}\ (1)] + 0\ [\text{if}(x_2 > 2.5),\ \text{then}\ (6)]$$

$$= 0.95 + 0.2 \times 1 + 0.4 \times 3 + 0 \times 2 + 0.3 \times 1 + 0 \times 6 = 2.65 \quad (3)$$

Conditions in the statements only take descriptive attributes into account because the rules will be applied to the new unlabeled instances. In this paper, there are eight target variables, and thus each rule will give a resultant vector of dimension eight. The adopted MTRM is PCTs employing random forests, and the model is transcribed into a rule ensemble for better interpretation, enabling the proposed model to predict multiple targets simultaneously.

2.3. Clus

MTRM has been implemented in the Clus, an open-source machine learning software that can be downloaded from [28]. Clus is a decision tree and rule learning system that works in PCTs [14]. It is a Java-based platform to build both classification and regression trees by choosing different operation settings. It has been successfully applied to plenty of tasks, including multi-target regression and classification, structured output learning, time series prediction, etc [14]. Clus provides many choices for operation settings. In particular, the operation settings related to the multiple-target regression are explained. First, three input files are required: (1) a file with training data, (2) a file with test data, and (3) a file specifying all the parameter settings. The training and test data dictionary (i.e., files names and variable types) should be listed in these setting files. Descriptive and target attributes in the dataset should be specified explicitly. Other functionalities, including choices of ensemble method and rule ensemble, should be addressed accordingly. Appendix A presents a brief example of input files. Full practical example files are available in [29]. After training the model, an output file will be generated which contains predictions for multiple target attributes. In addition, one can access the graphic PCTs in the output file of which example is shown in Appendix B. One is referred to the Clus manual for detailed instructions and additional settings.

3. Prediction of capacity curve

Although the proposed ML-based approach to capacity curve prediction can be applied to any RC structure, this study demonstrates the potential by focusing on rectangular RC shear walls' capacity curves. The training database is built upon a hybrid database consisting of real experimental results and computational simulation results. A high-prediction parallel finite element analysis platform (called VEEL, meaning virtual earthquake engineering laboratory) has been adopted to ensure reliably simulated curves. VEEL's general applicability and accuracy have been well documented in [30]. VEEL is rooted in a number of microphysical mechanisms, including a multi-directional smeared crack model, a topological information-based steel bar model capable of capturing progressive bar buckling, a 3D interlocking-based nonlinear shear

mechanism, and a bar-concrete proximity-based general confinement model. An optimized parallel computing algorithm is leveraged to effectively link millimeter length-scale mechanisms to real-scale RC walls [31,32].

3.1. Transform capacity curve into multivariate targets

The size of the experiment-based database is too small for ML training. We need to enrich the experimental database with simulated data without introducing a substantial loss of accuracy. The original database contains global F-D responses of seven rectangular shear walls (i.e., RW1, WSH1, WSH2, WSH3, WSH4, WSH5, and WSH6). The contrast between experimental F-Ds from existing literature [2,33] to F-Ds simulated by VEEL is performed in Fig. 2 to emphasize the precision of the original database. As summarized in Table 4, the variances occur in the axial force ratio (a_f) in percentage, yield stress (f_y) in MPa, the diameter of vertical reinforcement (d_b) in millimeter, and concrete compressive strength (f'_c) in MPa. It is challenging to rephrase the continuous capacity curve into the multivariate target, which machine learning can learn and predict. The overall procedures to extract the F-D capacity curve database are illustrated in Fig. 3. In Task 1 of Fig. 3, it is essential to extract the outermost points. Most of the outermost points are related to the overall envelope of the capacity curve of a shear wall subjected to reverse and cyclic loading. Although there is no strict restriction, 46 points are extracted from the shear wall database, as visualized in Fig. 4. More points will improve the accuracy of the fitted capacity curve, but this choice appears acceptable to capture the overall nonlinear envelops reasonably. The extracted points on the capacity curve envelope are denoted as $\{d_i, F_i\}, i = 1, \dots 46$, where d_i is a displacement and F_i is the associated force point. We perform separate least-square fittings on the positive and negative regimes to account for asymmetric shapes of general capacity curve envelopes. $\boldsymbol{\beta} \in \mathbb{R}^p$ stands for parameters to be determined, and $\boldsymbol{\beta} = [\boldsymbol{\beta}_P : \boldsymbol{\beta}_N]$, $\boldsymbol{\beta}_P = \{P_1, P_2, \dots, P_p\}^T$ and $\boldsymbol{\beta}_N = \{N_1, N_2, \dots, N_p\}^T$. Then, the optimal parameters (denoted by $\widehat{\boldsymbol{\beta}}$) for the positive and negative regimes are obtained by

$$\widehat{\boldsymbol{\beta}}_P = \underset{\boldsymbol{\beta}_P}{\operatorname{argmin}} \left\| \boldsymbol{F} - \mathbf{d}\boldsymbol{\beta}_p \right\|^2, \text{for } d_i \in \mathbb{R}^+ \tag{4}$$

$$\widehat{\boldsymbol{\beta}}_N = \underset{\boldsymbol{\beta}_N}{\operatorname{argmin}} \left\| \boldsymbol{F} - \mathbf{d}\boldsymbol{\beta}_n \right\|^2, \text{for } d_i \in \mathbb{R}^- \tag{5}$$

where $\mathbf{d}$ is the model matrix, $\mathbf{d} \in \mathbb{R}^{46\times4}$ of which i_{th} row means $\boldsymbol{d}_i = \{d_i, d_i^2, d_i^3, d_i^4\}$. The envelope force vector is $\boldsymbol{F} = \{F_1, F_2, \dots, F_{46}\}$. Thus, the p-parameter fitted model for the capacity curve envelop is succinctly given by:

$$F_i = H(d_i) \sum_{l=1}^{p} P_l d_i^l + H(-d_i) \sum_{l=1}^{p} N_l d_i^l \tag{6}$$

where $H(d)$ is the unit step function (i.e., one for $d > 0$, zero otherwise); p is the highest order of base polynomials. This study chose $p = 4$ for the polynomial bases rooted in the prior knowledge that most capacity curves often exhibit convex or concave shapes. A higher-order fitting may help, but our choice is justifiable since the values of R^2 (the coefficient of determination) calculated using our approach are commonly larger than 0.99. For the subsequent multi-target machine learning, we added the optimal parameters

$\widehat{\boldsymbol{\beta}} = [\widehat{\boldsymbol{\beta}}_p : \widehat{\boldsymbol{\beta}}_n] = \{\hat{P}_1, \hat{P}_2, \hat{P}_3, \hat{P}_4, \hat{N}_1, \hat{N}_2, \hat{N}_3, \hat{N}_4\}^{\mathrm{T}}$ onto the existing wall database. Thus, 32 descriptive variables and eight target variables are included in the finalized database. Detailed variable information is summarized in Appendix C. Overall, the capacity curve database dimension is 182×40 (i.e., 182 instances with 40 attributes).

Table 4
Details of the original rectangular shear wall database.

	RW1	WSH1	WSH2	WSH3	WSH4	WSH5	WSH6
a_f	0 ~ 30	0 ~ 40	0 ~ 40	0 ~ 40	0 ~ 40	0 ~ 40	0 ~ 40
f_y	300 ~ 600	450 ~ 610	500 ~ 710	500 ~ 720	500 ~ 640	500 ~ 710	500 ~ 650
d_b	12.7 ~ 28.6	8 ~ 14	8 ~ 15	8 ~ 15	8 ~ 15	6 ~ 12	8 ~ 15
f'_c	37.7	30 ~ 60	30 ~ 60	30 ~ 60	30 ~ 60	30 ~ 60	30 ~ 60

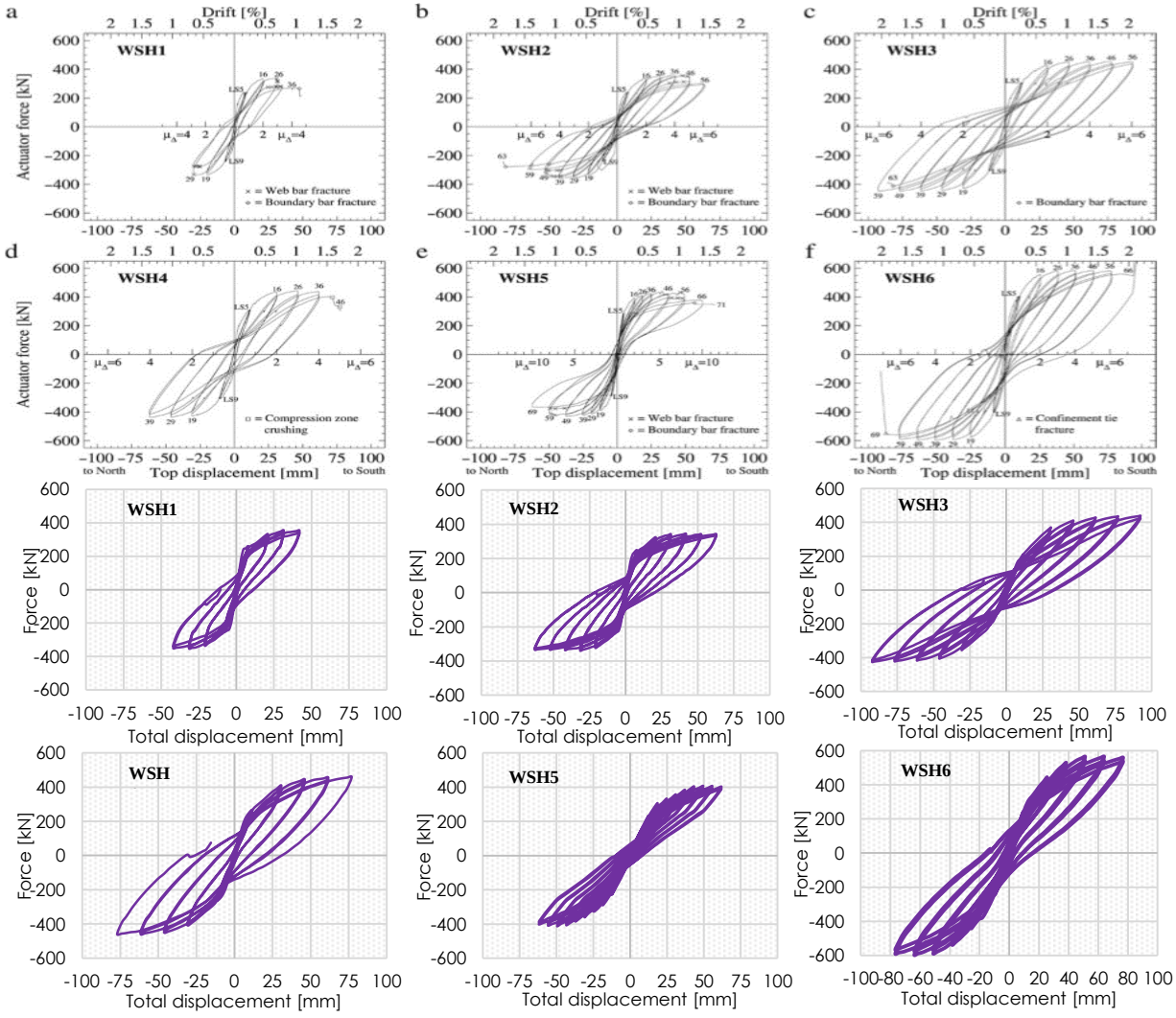


Fig. 2. (Top six panels) experimental F-D responses versus (bottom six panels) simulated F-D responses by VEEL.

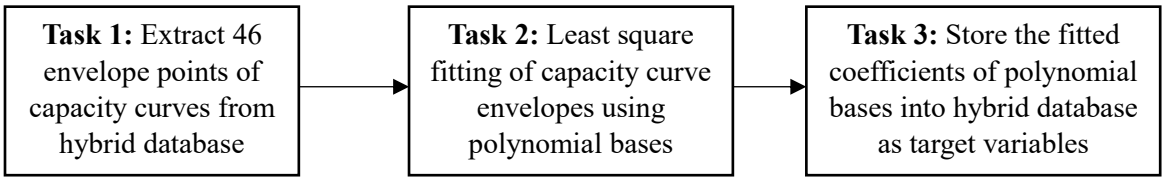


Fig. 2. Flowchart of transformation of capacity curve database to multiple target database.

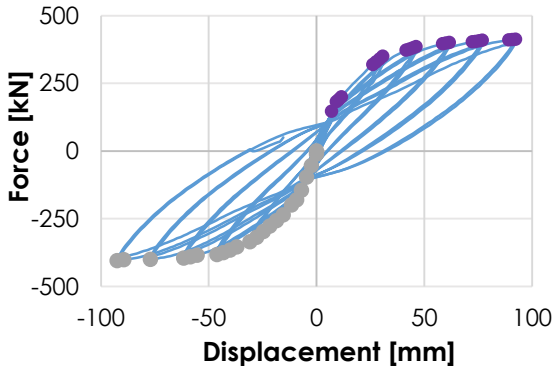


Fig. 3. Example of extraction of 46 outermost points from force-displacement (F-D) responses.

3.2. Multi-target prediction of capacity curve

This section explains the complete process of the multiple target ML prediction of the capacity curves using PCTs. PCTs consider trees as a hierarchy of clusters with respect to many observed descriptive variables to build trees to predict multiple targets simultaneously. As explained in the previous section, our hybrid database contains 32 descriptive variables (denoted as $\mathbf{X} \in \mathbb{R}^{n\times 32}$) and eight target variables ($\mathbf{Y} \in \mathbb{R}^{n\times 8}$). Thus, the i_{th} row of $\mathbf{X}$ is $\boldsymbol{x}_{(i)} = \{x_1, \dots, x_{32}\}_{(i)}$ whereas the i_{th} row of $\mathbf{Y}$ is $\{\hat{P}_1, \hat{P}_2, \hat{P}_3, \hat{P}_4, \hat{N}_1, \hat{N}_2, \hat{N}_3, \hat{N}_4\}_{(i)}$. The prediction task is to predict $\boldsymbol{y}_{(new)} \in \mathbb{R}^8$ given a new query of $\boldsymbol{x}_{(new)} \in \mathbb{R}^{32}$. Fig. 5 summarizes general procedures of initial setup, training, prediction, and visualization. We will elaborate on each sub-task as follows.

3.2.1. Initial preparation

Task 1 in Fig. 5 summarizes the key procedure before launching multiple target ML. Ranges of variables in the hybrid database are wide, e.g., ranging from 0.01 to 2.23×10^9. To be consistent and prevent any unit-dependent effect in PCTs, we normalized all attributes to the range of [0, 1]. We considered two normalization schemes: "min-max" and "standard deviation" normalizations as candidates. In the min-max normalization, normalization is done by

$$x_i' = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{7}$$

where x_{min} and x_{max} are the minimum and maximum of the i_{th} attribute, respectively. In the standard deviation normalization, we have

$$x_i' = \frac{x_i - \bar{x}}{s} \tag{8}$$

where $\bar{x}$ and s is the mean and the standard deviation of the i_{th} attribute, respectively. To quantitatively compare impacts of the normalization schemes, we compare multi-target

predictions of three cases: using (1) the original database without any normalization, (2) database normalized by the min-max scheme, and (3) database normalized by the standard deviation. All the initial settings of the MTRM model are constrained identical for three cases. From this preliminary comparison, the "min-max normalization" appears to lead to the lowest *MAE*.

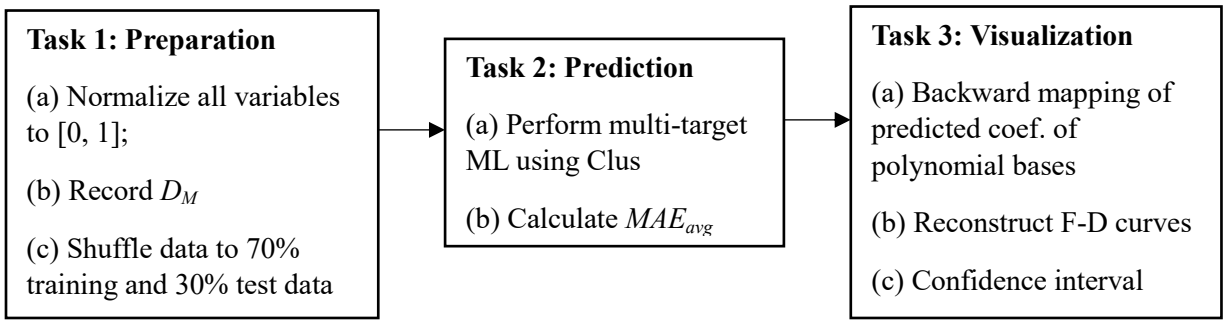


Fig. 4. Multi-target prediction flowchart from initial preparation, training and prediction, and postprocessing and investigation. (D_M: Mahalanobis Distance; MAE_{avg}: the averaged mean absolute error of the multiple-target prediction).

Hence, this study adopts the "min-max" normalization throughout the following procedures. x_{max} and x_{min} of each attribute must be stored for future backward mapping (i.e., from the normalized target to actual response, Task 2 (b) of Fig. 5).

Although our hybrid dataset has more than 200 instances, it is still relatively small for reliable ML training. The PCTs may not be stable to learn the rules around the outside borders of multiple descriptive variables. Such an issue is the so-called "extrapolation" problem, an intrinsic statistical model. In short, a statistical or ML model can predict well when the new instance is similar to those inside the data space. Still, its accuracy decreases as the new instance is near the borderlines or beyond the data space. In those ranges, prediction becomes an extrapolation since similar cases have never been experienced [34]. Therefore, it is important to understand each instance's relative location in the entire data space. In addition, it is instructive to note that the data space covered by the database is scattered and refers to space with more than one instance experienced inside. In the hope of quantitatively determining the borderlines of scattered data space and facilitating visualization of the relative position of new instances in the entire data space, we adopted the Mahalanobis Distance (denoted as D_M). For a data point in the multidimensional space, D_M measures how many standard deviations away the point is from the mean of the multidimensional space by

$$D_M(\boldsymbol{x}) \equiv \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^T \mathbf{S}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})} \tag{9}$$

where $\boldsymbol{x}$ is an instance in the descriptive data space (here $\boldsymbol{x} = \{x_1, x_2, \dots, x_{32}\}^T$), $\boldsymbol{\mu}$ is a vector of the mean of each descriptive variable (here, $\boldsymbol{\mu} = \{\mu_1, \mu_2, \dots, \mu_{32}\}^T$) and $\boldsymbol{S}$ is the covariance matrix. We calculate and record D_M into the database as auxiliary information (Task 1 (b) of Fig. 5). This information determines whether new data is inside the database space or close to or beyond the existing database. To facilitate the unbiased training of PCTs, we randomly shuffled the database to make 70% training data and 30% test data (Task 1 (c) of Fig. 5).

3.2.2. Training and test of the multi-target prediction model

As shown in Task 2 of Fig. 5, the next step is to train and perform the multi-target prediction. PCTs generate two types of prediction results: original predictions and pruned predictions. A very large PCTs is grown, which typically learns the details and noises in the training data to the extent that it will negatively influence the performance of the model on new instances. The PCTs are pruned by one of the pruned criteria to eliminate the negative impact. Here, only original predictions are considered in this paper because the pruned prediction is only necessary for the very large data set [16]. Random forests are used as an ensemble learning method. Among many measurements of prediction accuracy in the ML domains, we adopted the mean absolute error (MAE). Since we are predicting multiple targets, each target has its own MAE by:

$$MAE_i = \frac{100}{n}\sum_{j=1}^{n}\left|\frac{A_{i(j)}-P_{i(j)}}{A_{i(j)}}\right| \tag{10}$$

where MAE_i is the MAE of the i_{th} target, $A_{i(t)}$ and $P_{i(j)}$ is the true value and predicted value of the i_{th} target of the j_{th} instance, respectively. n is the total number of instances. Then, the overall MAE of all target attributes (denoted as MAE_{avg}) is calculated as:

$$MAE_{avg} = \frac{1}{q}\sum_{i=1}^{q} MAE_i \tag{11}$$

where q is the number of total target attributes. In this study, q = 8 (see Task 2 (b) of Fig. 5).

3.2.3. Visualization of prediction mode

Task 3 of Fig. 5 summarizes the postprocessing. Since our target is to predict curves (not a simple scalar), we reconstruct the capacity curves using the predicted coefficients of the polynomial bases. It starts from the backward mapping of the coefficients from [0, 1] to the original ranges. Given the predicted matrix $\mathbf{Y}_{pred} \in \mathbb{R}^{n\times 8}$ with each entity ranging [0, 1], a batch backward mapping is simply given by

$$\mathbf{Y}_{final} = \mathbf{Y}_{pred}\mathbf{Y}_{diff} + \mathbf{Y}_{min} \tag{12}$$

where $\mathbf{Y}_{pred} \in \mathbb{R}^{n\times 8}$ is the final predicted coefficient matrix with original ranges. $\mathbf{Y}_{diff} \in \mathbb{R}^{8\times 8}$ is a diagonal matrix and $\mathbf{Y}_{min} \in \mathbb{R}^{n\times 8}$ is a column-size identical matrix, which is given by

$$\mathbf{Y}_{diff} \equiv \begin{bmatrix} (\max(\boldsymbol{y}_1) - \min(\boldsymbol{y}_1)) & & [\mathbf{0}] \\ & \ddots & \\ [\mathbf{0}] & & (\max(\boldsymbol{y}_8) - \min(\boldsymbol{y}_8)) \end{bmatrix}$$

$$\mathbf{Y}_{min} \equiv \begin{bmatrix} (\min(\boldsymbol{y}_1)) & \cdots & (\min(\boldsymbol{y}_8)) \\ || & \ddots & || \\ (\min(\boldsymbol{y}_1)) & \cdots & (\min(\boldsymbol{y}_8)) \end{bmatrix}$$

Here $\boldsymbol{y}_i \in \mathbb{R}^{n\times 1}$ represents a vector of original i_{th} target coefficient. Since we now have all coefficients of the polynomial bases, we can draw the envelopes of the capacity curves by using Eq. (6).

3.2.4. Confidence interval

As all statistical models involve uncertainty, our multiple-target prediction model naturally exhibits uncertainty for new predictions. For a new prediction, it is crucial to provide uncertainty that is rooted in the training process that uses randomly selected training data sets. To offer a measurement of uncertainty behind ML-based prediction, this study harnesses a bootstrapping [34] similar to the so-called "percentile bootstrapping." The detailed procedure to obtain bootstrapping sample is as follows.

[BS 0] Initial stage begins with a training data set $M_{(i=1)}$ and a new instance $\boldsymbol{x}_{new} \in \mathbb{R}^{32\times1}$

[BS 1] Fit a multiple-target prediction model using the training data set $M_{(i)}$ and obtain a target response $\boldsymbol{y}_{new(i)} = \{P_1, P_{2,} P_3, P_4, N_1, N_2, N_3, N_4\}_{(i)}^{\mathrm{T}}$ for the given $\boldsymbol{x}_{new}$.

[BS 2] Generate a new training data set $M_{(i+1)}$ by resampling 70% of the database (randomly selected with replacement).

[BS 3] Refit the multiple-target prediction model using the training dataset $M_{(i+1)}$.

[BS 4] Repeat above steps (**1-3**) n_{bs} times to generate n_{bs} bootstrapping samples (i.e., n_{bs} multi-target predictions).

In our approach, sorting the n_{bs} multi-target predictions is necessary, but it is not straightforward as a single target bootstrapping. To derive a physically sound approach for sorting the n_{bs} multivariate predictions, we focused on the absorbed energy of the structure, i.e., area under the capacity curves. In general, a peak-based sorting appears not reasonable: e.g., curve (c) has the largest positive peak while curve (a) has the largest peak in the negative regime in Fig. 6. However, the total absorbed energy intuitively leads to a single scalar that also holds the mechanical meaning of the structure. Fig. 6 briefly illustrates how the capacity curves' absorbed energy is calculated and how it can help order the three dissimilar curves of different peaks and shapes. Since we represent the capacity curve envelopes with polynomial bases and already obtained their real-valued coefficients in $\boldsymbol{y}_{new(i)}, i = 1, \dots, n_{bs}$ (**BS 2**), it is straightforward to calculate the absorbed energy (denoted as $I_{(i)} \in \mathbb{R}^{+}$) as

$$I_{(i)} = \left| \int_{D_{min,(i)}}^{D_{\max,(i)}} H\left(\zeta_{(i)}\right) \sum_{l=1}^{p} P_l \zeta_{(i)}^{l} + H\left(-\zeta_{(i)}\right) \sum_{l=1}^{p} N_l \zeta_{(i)}^{l} \; d\zeta_{(i)} \right| \tag{13}$$

where the subscript (i) denotes the i_{th} multi-target prediction; |.| returns the absolute value; $H(d)$ is the unit step function (i.e., 1.0 for $d > 0$, zero otherwise); $D_{max,(i)}$ and $D_{min,(i)}$ is the positive maximum and negative minimum displacement of the capacity curve, respectively; $\zeta_{(i)}$ is the displacement coordinate. The condition that the cumulative distribution of bootstrap samples (denoted as $\hat{G}$) is less than or equal to a constant b is expressed as:

$$\hat{G}(b) = F\{I_{(i)} \leq b\}, i = 1, \dots, n_{bs}. \tag{14}$$

where F is the frequencies of $I_{(i)}$. An instance with a specific percentile (α) is represented as:

$$\boldsymbol{y}^{*(\alpha)} = \hat{G}^{-1}(\alpha) \tag{15}$$

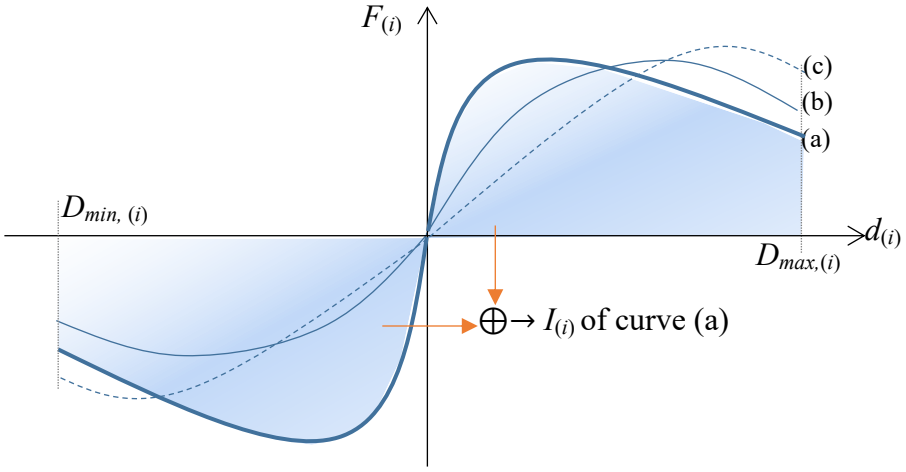


Fig. 5. Illustration of calculation of the absorbed energy used for sorting in bootstrapping. Three capacity curves (a,b,c) with different peaks and shapes are shown. (⊕ means a summation operation).

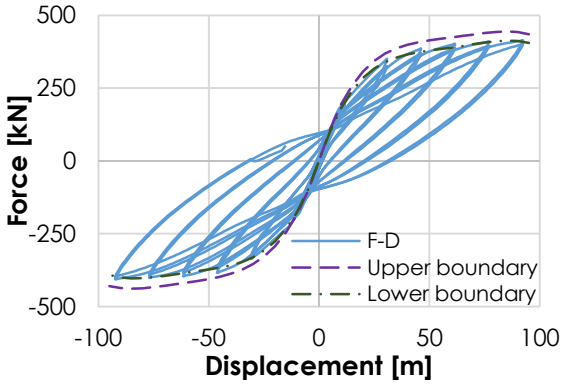


Fig. 7. 95% percentile confidence interval of wall WSH3 under 590 MPa shear strength.

where $\hat{G}^{-1}$ is the inverse function of $\hat{G}$. Therefore, the 95% percentile confidence interval is given by

$$\left(\boldsymbol{y}^{*(0.025)}, \boldsymbol{y}^{*(0.975)}\right) \tag{16}$$

In this paper, n_{bs} = 100 is adopted. Here, a 95% confidence interval indicates the probability of the range covering the predicted curves regarding the total absorbed energy. For instance, Fig. 7 shows a 95% percentile confidence interval. Note that there is ample room for extension of the proposed approach, especially regarding how to define the "order" of the bootstrapped samples. Also, there are other methods for uncertainty quantification, such as a Jackknife method [35], which is straightforward and does not require a random sampling.

4. Results

4.1. Impact of PCT types on prediction accuracy

To investigate the impact of PCT types on the performance of MTRM, we considered two types of operational settings of PCTs. The first type, conventional PCTs, considers both descriptive variables $\boldsymbol{X}$ and target attributes $\boldsymbol{Y}$ to partition instances into subsets during searching. On the contrary, the second type, the so-called trial PCTs, partitions instances into subsets in terms of only the descriptive variables $\boldsymbol{X}$. For comparison, we used identical training and test data from the capacity curve database (182 rows) to train the model and make predictions. As already mentioned in Task 1 (b) in Fig. 5, D_M of all instances are recorded to easily visualize each specimen's relative position in the multivariate space and are plotted in a radar plot (e.g., Fig.8). The detailed values of the created database and D_M of all instances are available in [29]. Four selected walls (indexed by 4, 20, 67, and 88) and their D_M values are presented in Fig. 8. Fig. 9 presents the predicted capacity curves of selected walls accordingly. The corresponding *MAE*s of these four capacity curves predicted by the conventional PCTs and trial PCTs are aggregated in Table 5.

Table 5
*MAE*s of prediction by conventional PCTs versus trial PCTs.

Wall Index	D_M	*MAEs* (conventional)	*MAEs* (Trial)
4	17.9	2.6%	4%
20	16.8	2.3%	18%
67	0.62	1.1%	1.2%
88	1.79	0.1%	0.1%

Another prediction of wall 175 with $D_M = 1.89$ is plotted in Fig.10, which also supports the good prediction of both PCTs with smaller D_M. The prediction accuracy of conventional PCTs is much stable and superior to the trial PCTs. In addition, it is observed that both conventional PCTs and trial PCTs make a relatively accurate prediction of wall index 67 and 88, but a decent prediction of wall index 4 and 20. To some extent, the trial PCTs is similar to the "clustering" since it considers only the descriptive attributes. On the contrary, the conventional PCTs collaborate with the rule ensemble to better interpret and explore complex data. In view of the high dimensionality of our database (i.e., 32 variables), the conventional PCTs appear to slightly outperform the trial PCTs. Based on this outcome, the conventional PCTs were utilized in all the simulations hereafter.

4.2. Impact of the extended database on the prediction

The discussion addressed so far is inherently based on the training data. It is common sense that PCTs will yield better predictions when a target instance resides within the boundary of the available training data. The prediction model will perform the so-called "extrapolation" when a new target has little similarity and falls outside the existing training data. To investigate the influence of this extrapolation, we first trained the PCTs with 70% of sampled training data from the capacity curve database (182 rows) and made the prediction for the 30% test data plus a new instance (SW1-2) inclusively involved. The D_M of SW1-2 along with other 182 instances are visualized in Fig. 11 (a). The D_M of SW1-2 (marked as a star) indicates exclusion of the new instance in contrast with the existing training data space. And the predicted capacity curve of SW1-2 is visualized in Fig.12 as the dashed curve. Secondly, we collected 33 new rectangular

shear walls from [36] and merged them into the capacity curve database, enlarging them to 214 rows. Repeat the scenario by training the PCTs with 70% of sampled training data from the extended capacity curve database (214 rows), and predict the rest of the test data.

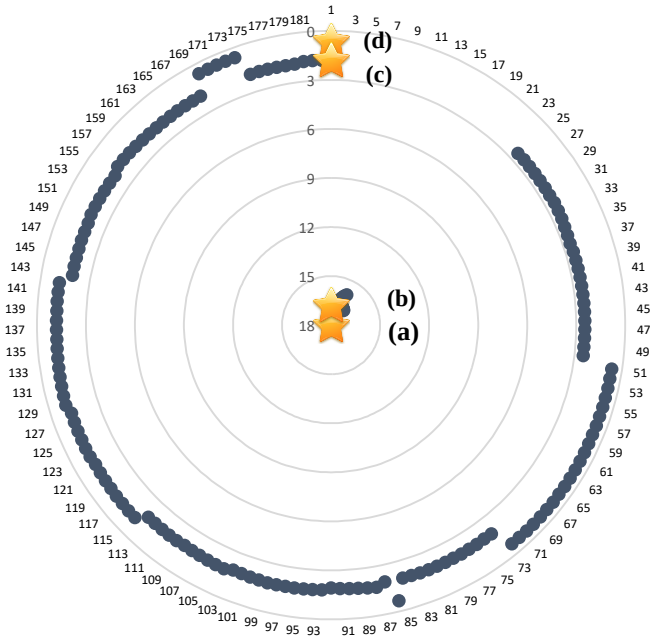


Fig. 8. Radar plot of 182 walls with varying D_M: (a) wall 4 with D_M = 17.99; (b) wall 20 with D_M = 16.76; (c) wall 67 with D_M = 0.62; (d) wall 88 with D_M = 1.79.

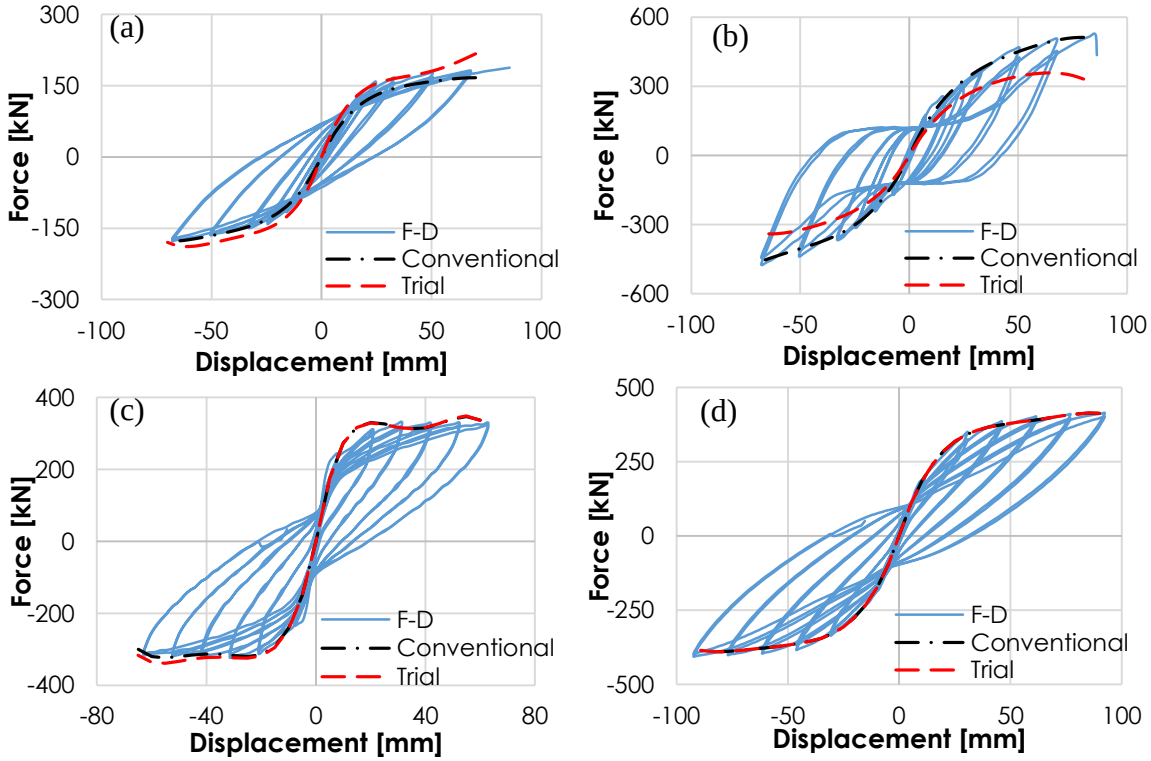


Fig. 9. Predicted capacity curves using the conventional PCTs and the trial PCTs: (a) wall 4; (b) wall 20; (c) wall 67; (d) wall 88.

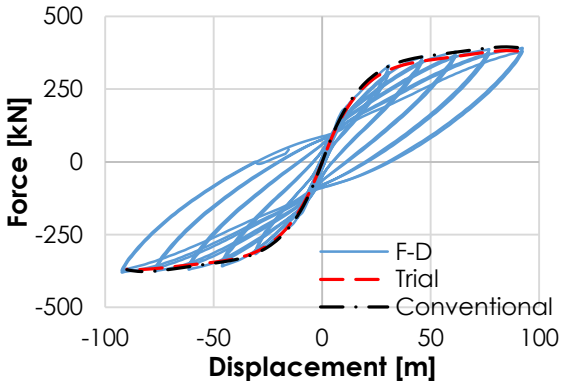


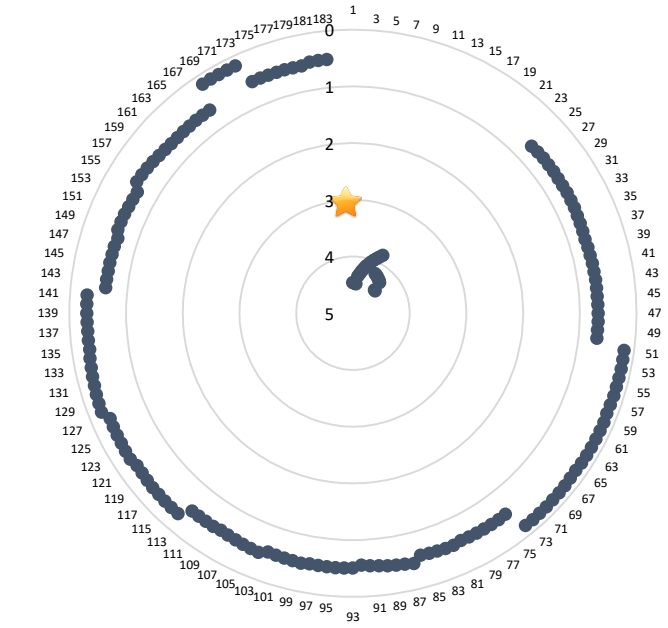
Fig. 10. Predicted capacity curves of wall 175 using the conventional PCTs and the trial PCTs.

It is critical to note that we force SW1-2 as one of the test data for both scenarios for comparison. The predicted capacity curve of SW1-2 is visualized as the dashed curve delimited by dots in Fig. 12. For the first scenario, it is observed that the prediction of SW1-2 under the original database diverges from the experimental F-D of SW1-2 in Fig. 12. And the data space of SW1-2 marked as star indicates exclusion of the new instance in contrast with the existing training data space in Fig. 11 (a). For the second scenario, we found that the prediction of SW1-2 under the extended database converges significantly in contrast with the prediction of the previous scenario. The ample data space around SW1-2 in Fig. 11 (a) has been compacted asymptotically with multiple samples around in Fig. 11 (b), which presents that the extension of 33 new shear walls has high similarity with specimen SW1-2 in terms of D_M. Analyzing the results of both scenarios, the extension of the database, which includes instances of high similarity with SW1-2 in terms of D_M, will positively influence the prediction. These similar instances will fill in scattered data space around SW1-2 and lead to a more comprehensive model. On the contrary, an extension of the database of low similarity with SW1-2 will rarely promote the prediction of SW1-2.

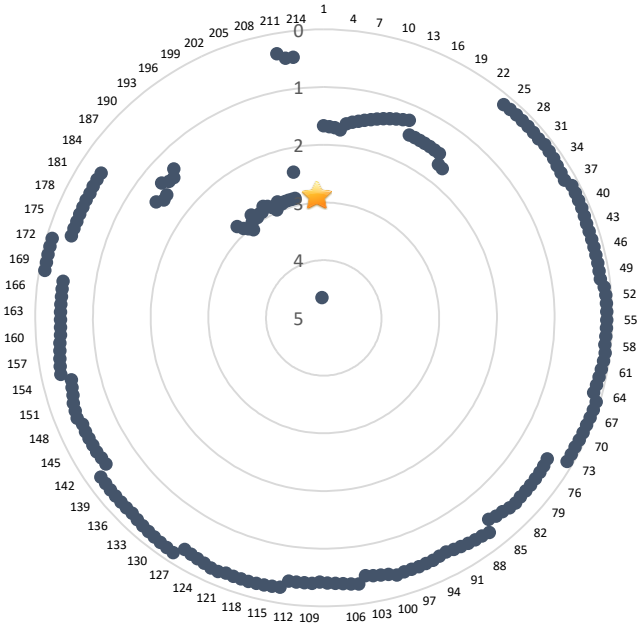
4.3. Impact of erroneous data on prediction

Nowadays, a tremendous amount of engineering data accumulate in our domain, frequently reported with incompleteness and erroneous values. The deficiency of data will not facilitate training of the predictive models but leave the potential risk of generating an unstable prediction. One possible way to minimize the negative influence of data deficiency on prediction is to leverage imputation to handle missing values. The impacts of the existing imputation method fractional hot deck imputation on the prediction of engineering data have been investigated by [37]. The robustness of the MTRM against erroneous data is one of the most important criteria to evaluate the model objectively. Note that the naive version of the capacity curve database (182 rows) has 2.3% erroneous values within the descriptive variable matrix $\boldsymbol{X}$ because of human errors. Fortunately, the author was aware of these errors ahead of time and remedied the capacity curve database with extreme caution. To investigate the impact of erroneous data on the prediction, the author utilized 30% of sampled erroneous data to train the conventional PCTs and generate predictions for wall indexed by 20 and 88, respectively. (the identical walls denoted

by (b) and (d) in Fig. 8). The predicted F-D curves of these two targets are visualized in Fig. 13 as dash curves in contrast with predicted capacity curves upon the correct database (dash curves delimited with dots). Fig. 13 infers that the conventional PCTs are fairly robust against erroneous data.



(a)



(b)

Fig. 11. (a) D_M of 183 wall instances (b) D_M of 214 wall instances. Note that D_M of wall SW1-2 is marked with a star.

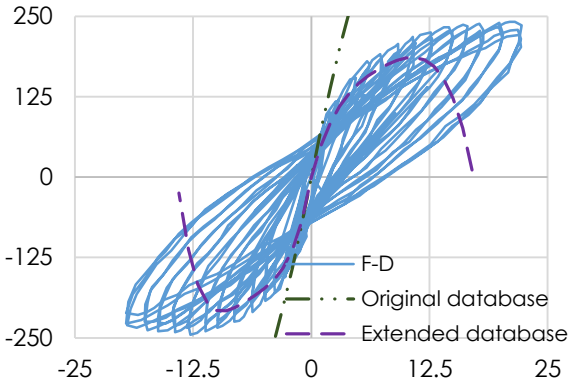


Fig. 12. The predicted capacity curves of SW1-2 based on the original database and the extended database, respectively.

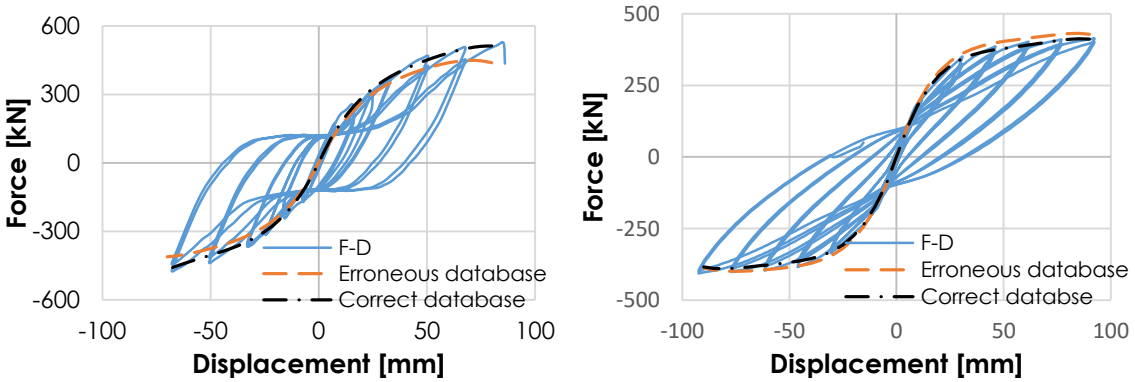


Fig. 13. The predicted capacity curves based on erroneous database versus that based on the correct database: (a) wall 20; (b) wall 88.

5. Conclusions

In the hope of providing an efficient and reliable tool that can help quickly determine the capacity curves of F-D responses, this paper utilizes a multi-target regression model to generate the prediction. To our best knowledge, the prediction of capacity curves had never been attempted in infrastructure engineering. The general conclusion is that the MTRM implementing conventional PCTs combined with ensemble methods generates fairly good predicted F-D curves in terms of *MAE* and visualization. Its confidence interval and robustness against erroneous data strengthen the reliability of the method. Compared with the traditional approach to conducting a real experiment or simulating finite element models, the proposed method of incorporating ML will significantly reduce expenses in terms of time and money.

The future works will focus on several interesting aspects which will promote the performance of the method. Firstly, the university of the proposed capacity curve database is restricted to rectangular shear walls. The extension of the method to other types of infrastructures will break the bottleneck of the proposed approach. Secondly, the capacity curve database consists of 32 descriptive variables currently, which may cause overfitting issues. An attributes selection test based on empirical engineering knowledge or the attributes selection algorithm [34] may improve the precision of the prediction. Lastly, concerning the size of the proposed capacity database, it may result in unstable and biased models. A sufficiently large database extended in the future will help to produce more accurate and stable results.

Acknowledgments

The authors declare no conflict of interest. This research is supported by the research funding of the Department of Civil, Construction, and Environmental Engineering of Iowa State University. The research reported is partially supported by the HPC@ISU equipment at ISU, some of which has been purchased through funding provided by NSF under MRI grant number CNS 1229081 and CRI grant number 1205413.

Appendix A: Example of input files

The user must provide three input files to run the Clus. The training data file should strictly follow the format:

```
@RELATION          "WallDB_train"

@ATTRIBUTE         var1            numeric
@ATTRIBUTE         var2            numeric
      ⋮             ⋮                ⋮
@ATTRIBUTE         var40           numeric

@DATA
0,0,0,0,1,0.25,0,0.256666667,0,0,0.048,0,0.226190476,5.07E-08,1,1,0.155506608,0,
                                    ⋮
```

The Clus is format-sensitive. The exact name of the training data file should be included at the beginning. Afterward, users have to list all attributes along with data types. Note that training data must be listed row-wise. A comma delimits each element in a row, and each row is delimited by starting a new line. The test data file follows an identical format. Besides, a file specifying all the parameter settings is described as:

```
[Attributes]                    [Ensemble]
Target = 33-40                  Iterations = 100
Clustering = 1-40               EnsembleMethod = RForest
Descriptive = 1-32

[Data]                          [Output]
File = WallDB_train.arff        WritePredictions = {Train,Test}
TestSet = WallDB_test.arff

[Tree]
Heuristic = VarianceReduction
PruningMethod = M5Multi
ConvertToRules = ALLNodes
```

Users can control the types of PCTs in Attributes section. Data section lists the full name with the extension of training data and test data. Tree and Ensemble sections specify additional settings for the PCTs. For more details, Clus manual provides comprehensive explanations for each item in these three files.

Appendix B: Graphic PCTs

This appendix explains a toy graphic PCTs after rule ensemble in Fig. 1 (b). An incomplete realistic graphic PCTs for predictions in Figs. 9 and 10 is obtained from the output file of Clus:

```
Original Model
**************

var1 > 0.0
+--yes: var14 > 0.0
|       +--yes: var16 > 0.569029851
|       |       +--yes: var7 > 0.545454545
|       |       |       +--yes: var6 > 0.5
|       |       |       |       +--yes: [0.307979,0.527636,0.6267,0.962626,0.756192,0.477687,0,0.26211]: 3
|       |       |       |       +--no:  var8 > 0.276666667
|       |       |       |               +--yes: [0.240821,0.661636,0.422535,0.971888,0.796779,0.588153,0,0.400133]: 2
|       |       |       |               +--no:  var6 > 0.125
|       |       |       |                       +--yes: var13 > 0.80952381
|       |       |       |                       |       +--yes: var17 > 0.088105727
|       |       |       |                       |       |       +--yes: [0.253708,0.632909,0.479085,0.967333,0.789692,0.563855,0,0.309887]: 2
|       |       |       |                       |       |       +--no:  [0.259163,0.624,0.489324,0.966503,0.7865,0.557831,0,0.301482]: 3
|       |       |       |                       |       +--no:  [0.253708,0.633818,0.476831,0.966779,0.788548,0.562249,0,0.297279]: 2
|       |       |       |                       +--no:  [0.253731,0.630737,0.497282,0.961661,0.788452,0.559556,0,0.199005]: 2
|       |       |       +--no:  var6 > 0.5
|       |       |               +--yes: [0.698068,0.250182,0.568923,0.990174,0.310385,0.231727,0,0.881221]: 2
|       |       |               +--no:  var8 > 0.35
|       |       |                       +--yes: [0.210789,0.726788,0.310329,0.977614,0.816987,0.652209,0,0.509401]: 3
|       |       |                       +--no:  var6 > 0.1425
|       |       |                               +--yes: [0.199034,0.76,0.219718,0.989024,0.834487,0.704016,0,0.757133]: 3
|       |       |                               +--no:  var17 > 0.220264317
|       |       |                                       +--yes: var17 > 0.308370044
|       |       |                                       |       +--yes: [0.214614,0.718727,0.321127,0.979046,0.81375,0.643574,0,0.551427]: 2
|       |       |                                       |       +--no:  [0.220048,0.708182,0.337324,0.977094,0.809904,0.633936,0,0.510949]: 2
|       |       |                                       +--no:  var8 > 0.166666667
|       |       |                                               +--yes: var13 > 0.619047619
|       |       |                                               |       +--yes: var17 > 0.13215859
|       |       |                                               |       |       +--yes: [0.219722,0.7079,0.345352,0.974588,0.807923,0.6249,0,0.429993]: 4
|       |       |                                               |       |       +--no:  [0.197343,0.753818,0.267606,0.979599,0.808654,0.628313,0,0.44791]: 2
|       |       |                                               |       +--no:  [0.203019,0.741636,0.288732,0.978395,0.809712,0.631325,0,0.457863]: 2
|       |       |                                               +--no:  [0.227295,0.694182,0.362676,0.974393,0.81125,0.635944,0,0.471798]: 2
|       |       +--no:  var14 > 1.3E-7
|       |               +--yes: var8 > 0.306666667
|       |               |       +--yes: var8 > 0.5
|       |               |       |       +--yes: var8 > 0.666666667
|       |               |       |       |       +--yes: [0.256256,0.633691,0.458451,0.972506,0.80174,0.595602,0,0.451228]: 2
|       |               |       |       |       +--no:  var6 > 0.375
|       |               |       |       |               +--yes: [0.271884,0.599182,0.512254,0.971107,0.775828,0.529707,0,0.374253]: 2
|       |               |       |       |               +--no:  var17 > 0.176211454
|       |               |       |       |                       +--yes: var6 > 0.125
|       |               |       |       |                       |       +--yes: var17 > 0.264317181
|       |               |       |       |                       |       |       +--yes: [0.259529,0.623364,0.484648,0.970651,0.782962,0.549538,0,0.388852]: 2
|       |               |       |       |                       |       |       +--no:  var13 > 0.833333333
|       |               |       |       |                       |       |               +--yes: [0.259227,0.624994,0.480376,0.970348,0.784808,0.555408,0,0.378898]: 3
|       |               |       |       |                       |       |               +--no:  [0.259541,0.624182,0.48331,0.969643,0.785067,0.556325,0,0.372263]: 2
|       |               |       |       |                       |       +--no:  [0.259179,0.624709,0.490845,0.965576,0.784038,0.553434,0,0.28069]: 2
|       |               |       |       |                       +--no:  [0.257907,0.628109,0.478638,0.968677,0.785635,0.558701,0,0.35324]: 3
|       |               |       |       +--no:  [0.262488,0.617218,0.500211,0.96704,0.783106,0.55006,0.5,0.318514]: 2
```

We upload the output file of predictions in Figs. 9 and 10 with full graphic PCTs in [29].

Appendix C: Attributes details of the capacity curve database

Attributes	Detail
I	Moment of inertia
Length	Length of shear wall
Thickness	Thickness of shear wall
Height	Height of shear wall
Number of floors	Number of floors
Axial Force Ratio	Axial force ratio
Cover thickness	Cover thickness
Concrete_fc	Concrete compressive strength
Concrete_ft	Concrete yield strength
bb	Width of boundary element
hb	Thickness of boundary element
cb	Cover thickness in boundary element
Steel_Vertical1_fy	Yield strength of boundary longitudinal reinforcement

Steel_Vertical1_fu	Ultimate stress of boundary longitudinal reinforcement
Steel_Vertical1_Spacing	Spacing of boundary longitudinal reinforcement
Steel_Vertical1_strain at fu	Ultimate strain of boundary longitudinal reinforcement
Steel_Vertical1_Diameter	Diameter of boundary longitudinal reinforcement
Steel_Vertical2_fy	Yielding strength of web longitudinal reinforcement
Steel_Vertical2_fu	Ultimate stress of web longitudinal reinforcement
Steel_Vertical2_Diameter	Diameter of web longitudinal reinforcement
Steel_Horizontal1_fy	Yielding strength of boundary transverse reinforcement
Steel_Horizontal1_fu	Ultimate stress of boundary transverse reinforcement
Steel_Horizontal1_strain at fu	Ultimate strain of boundary transverse reinforcement
Steel_Horizontal1_Spacing	Spacing of boundary transverse reinforcement
Steel_Horizontal1_Diameter	Diameter of boundary transverse reinforcement
Steel_Stirrup1_fy	Yielding strength of stirrups
Steel_Stirrup1_fu	Ultimate stress of stirrups
Steel_Stirrup1_strain at fu	Ultimate strain of stirrups
Steel_Stirrup1_spacing	Spacing of stirrups
Steel_Stirrup1_Diameter	Diameter of stirrups
Number of longitudinal bars at wall boundary	Number of longitudinal bars at wall boundary
$P_1 \sim P_p$ and $N_1 \sim N_p$	Polynomial bases parameters

References

[1] Aaleti S, Brueggen BL, Johnson B, French CE, Sritharan S. Cyclic Response of Reinforced Concrete Walls with Different Anchorage Details: Experimental Investigation. J Struct Eng 2013;139:1181–91. https://doi.org/10.1061/(ASCE)ST.1943-541X.0000732.

[2] Dazio A, Beyer K, Bachmann H. Quasi-static cyclic tests and plastic hinge analysis of RC structural walls. Eng Struct 2009;31:1556–71. https://doi.org/10.1016/j.engstruct.2009.02.018.

[3] Lefas ID, Kotsovos MD, Ambraseys NN. Behavior of Reinforced Concrete Structural Walls: Strength, Deformation Characteristics, and Failure Mechanism. ACI Struct J 1990;87:23–31. https://doi.org/10.14359/2911.

[4] Salonikios TN, Kappos AJ, Tegos IA, Penelis GG. Cyclic load behavior of low-slenderness reinforced concrete walls: design basis and test results. ACI Struct J 1999;96:649–60.

[5] Rafiq M., Bugmann G, Easterbrook D. Neural network design for engineering applications. Comput Struct 2001;79:1541–52. https://doi.org/10.1016/S0045-7949(01)00039-6.

[6] Reich Y. Machine learning techniques for civil engineering problems. Comput Civ Infrastruct Eng 1997;12:295–310.

[7] Chou J, Ngo N, Pham A. Shear Strength Prediction in Reinforced Concrete Deep Beams Using Nature-Inspired Metaheuristic Support Vector Regression. J Comput Civ Eng 2016;30:04015002. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000466.

[8] Pal M, Deswal S. Support vector regression based shear strength modelling of deep beams. Comput Struct 2011;89:1430–9. https://doi.org/10.1016/j.compstruc.2011.03.005.

[9] Aguilar V, Sandoval C, Adam JM, Garzón-Roca J, Valdebenito G. Prediction of the shear strength of reinforced masonry walls using a large experimental database and artificial neural networks. Struct Infrastruct Eng 2016;12:1661–74. https://doi.org/10.1080/15732479.2016.1157824.

[10] Chou J-S, Pham A-D. Enhanced artificial intelligence for ensemble approach to predicting high performance concrete compressive strength. Constr Build Mater 2013;49:554–63. https://doi.org/10.1016/j.conbuildmat.2013.08.078.

[11] van Gent MRA, van den Boogaard HFP. Neural network modelling of forces on vertical structures. Coast Eng Proc 1998:2096–123. https://doi.org/http://dx.doi.org/10.9753/icce.v26.%25p.

[12] Johari A, Habibagahi G, Ghahramani A. Prediction of Soil–Water Characteristic Curve Using Genetic Programming. J Geotech Geoenvironmental Eng 2006;132:661–5. https://doi.org/10.1061/(ASCE)1090-0241(2006)132:5(661).

[13] Saha S, Gu F, Luo X, Lytton RL. Prediction of Soil-Water Characteristic Curve Using Artificial Neural Network Approach. PanAm Unsaturated Soils 2017, Reston, VA: American Society of Civil Engineers; 2018, p. 124–34. https://doi.org/10.1061/9780784481684.014.

[14] Struyf J, Zenko B, Blockeel H, Vens C. Clus: user's manual. 2010.

[15] Ženko B. Learning predictive clustering rules. Int Work Knowl Discov Inductive Databases 2005;32:234–50.

[16] Blockeel H, De Raedt L, Ramon J. Top-down induction of clustering trees. Proc 15th Int Conf Mach Learn 1998:55–63. https://doi.org/10.1.1.50.3353.

[17] Prajapati P, Thakkar A. Improving the performance of predictive clustering tree algorithm for hierarchical multi-label classification. Proc. Int. Conf. Emerg. Res. Comput. Information, Commun. Appl., 2014.

[18] Jianhua Lin. Divergence measures based on the shannon entropy. IEEE Trans Inf Theory 1991;37:145–51.

[19] Karegowda AG, Manjunath AS, Ratio G, Evaluation CF. Comparative study of attribute selection using gain ratio. Int J Inf Technol Knowl Knowl Manag 2010;2:271–7.

[20] Opitz D, Maclin R. Popular Ensemble Methods: An Empirical Study. J Artif Intell Res 1999;11:169–98. https://doi.org/10.1613/jair.614.

[21] Rokach L. Ensemble-based classifiers. Artif Intell Rev 2010;33:1–39. https://doi.org/10.1007/s10462-009-9124-7.

[22] Tille Y. Statistical Analysis of Designed Experiments. vol. 53. New York: Springer-Verlag; 2002. https://doi.org/10.1007/b98966.

[23] Bbeiman LEO, Breiman L. Bagging predictors. Mach Learn 1996;24:123–40.

[24] Quinlan JR. Bagging, boosting, and C4.5. AAAI/IAAI, 1996, p. 725–30.

[25] Kocev D, Vens C, Strufy J, Dzeroski S. Ensembles of multi-objective decision trees. Mach Learn ECML 2007 2007:624–31.

[26] Aho T, Zenko B, Dzeroski S. Rule Ensembles for Multi-target Regression. 2009 Ninth IEEE Int. Conf. Data Min., IEEE; 2009, p. 21–30. https://doi.org/10.1109/ICDM.2009.16.

[27] Friedman JH, Popescu BE. Predictive learning via rule ensembles. Ann Appl Stat 2008;2:916–54. https://doi.org/10.1214/07-AOAS148.

[28] Struyf J. Clus download 2013.

[29] Yang Y, Cho IH. Shear wall database 2019.

[30] Cho IH. Virtual Earthquake Engineering Laboratory Capturing Nonlinear Shear, Localized Damage and Progressive Buckling of Bar. Earthq Spectra 2013;29:103–26. https://doi.org/10.1193/1.4000095.

[31] Cho IH, Hall JF. Parallelized Implicit Nonlinear FEA Program for Real Scale RC Structures under Cyclic Loading. J Comput Civ Eng 2012;26:356–65. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000138.

[32] Cho IH, Porter KA. Multilayered grouping parallel algorithm for multiple-level multiscale analyses. Int J Numer Methods Eng 2014;100:914–32. https://doi.org/10.1002/nme.4791.

[33] Thomsen JH, Wallace JW. Displacement-Based Design of Slender Reinforced Concrete Structural Walls—Experimental Verification. J Struct Eng 2004;130:618–30. https://doi.org/10.1061/(ASCE)0733-9445(2004)130:4(618).

[34] Song I, Cho I, Tessitore T, Gurcsik T, Ceylan H. Data-Driven Prediction of Runway Incursions with Uncertainty Quantification. J Comput Civ Eng 2018;32:04018004. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000733.

[35] Efron B, Stein C. The Jackknife Estimate of Variance. Ann Stat 1981;9:586–96. https://doi.org/10.1214/aos/1176345462.

[36] Tongji Univeristy. Database on static tests of structural members and joint assemblies. 2008.

[37] Song I, Yang Y, Im J, Tong T, Ceylan H, Cho IH. Impacts of Fractional Hot-Deck Imputation on Learning and Prediction of Engineering Data. IEEE Trans Knowl Data Eng 2020;32:2363–73. https://doi.org/10.1109/TKDE.2019.2922638.