computation

MDPI

Article

Response Spectrum Analysis of Multi-Story Shear Buildings Using Machine Learning Techniques

Manolis Georgioudakis [1,*] and Vagelis Plevris [2]

[1] Institute of Structural Analysis & Antiseismic Research, School of Civil Engineering, National Technical University of Athens, Zografou Campus, GR 15780 Athens, Greece
[2] Department of Civil and Environmental Engineering, Qatar University, Doha P.O. Box 2713, Qatar; vplevris@qu.edu.qa
* Correspondence: geoem@mail.ntua.gr

Abstract: The dynamic analysis of structures is a computationally intensive procedure that must be considered, in order to make accurate seismic performance assessments in civil and structural engineering applications. To avoid these computationally demanding tasks, simplified methods are often used by engineers in practice, to estimate the behavior of complex structures under dynamic loading. This paper presents an assessment of several machine learning (ML) algorithms, with different characteristics, that aim to predict the dynamic analysis response of multi-story buildings. Large datasets of dynamic response analyses results were generated through standard sampling methods and conventional response spectrum modal analysis procedures. In an effort to obtain the best algorithm performance, an extensive hyper-parameter search was elaborated, followed by the corresponding feature importance. The ML model which exhibited the best performance was deployed in a web application, with the aim of providing predictions of the dynamic responses of multi-story buildings, according to their characteristics.

Keywords: response spectrum analysis; ensemble algorithms; machine learning; shear building; SHAP explainability

Citation: Georgioudakis, M.; Plevris, V. Response Spectrum Analysis of Multi-Story Shear Buildings Using Machine Learning Techniques. *Computation* **2023**, *11*, 126. https://doi.org/10.3390/computation11070126

Academic Editors: Gavril Grebenisan, Alin Pop and Dan Claudiu Negrău

Received: 6 May 2023
Revised: 9 June 2023
Accepted: 20 June 2023
Published: 29 June 2023

1. Introduction

Machine learning (ML) has numerous applications in modeling and simulation of structures [1]. One of the most common applications of ML in structural analysis is the prediction of structural behavior under different loads and environmental conditions. ML algorithms can be trained on data from previous structural analyses, to learn how different factors—such as material properties, geometry, and loading conditions—affect structural response. This information can then be used to predict the behavior of new structures, without the need for time-consuming and expensive additional analyses. Another interesting field of application is structural health monitoring (SHM) and damage identification [2], where, by analyzing the changes in structural response over time, and using data collected by SHM systems, ML algorithms can learn to detect and localize damage in structures and, in general, assess the health and condition of a structure over time. In design optimization [3], by analyzing the relationships between different design parameters and structural performance, ML algorithms can identify optimal design configurations that minimize weight, maximize stiffness, or achieve other desired performance characteristics [4,5]. ML can be also used to quantify the uncertainties associated with structural analyses, improving the accuracy of predictions and reducing the risk of failure [6]. Overall, the use of ML in structural analysis has the potential to significantly improve the accuracy and efficiency of structural analysis, as well as to enable new capabilities for damage detection and design optimization.

In the specialized fields of structural dynamics and earthquake engineering, ML techniques are also increasingly being used [7], as they can help to extract useful insights

and patterns from large amounts of data, which can be used to improve the accuracy and efficiency of structural dynamic analysis techniques. Data-driven models for predicting the dynamic response of linear and non-linear systems are of great importance, due to their wide application, from probabilistic analysis to inverse problems, such as system identification and damage diagnosis. In the following paragraphs, we examine some important works and state-of-the-art contributions in the field.

Xie et al. [8] presented a comprehensive evaluation of the progress and challenges of implementing ML in earthquake engineering. Their study used a hierarchical attribute matrix to categorize the literature, based on four traits, (i) ML method; (ii) topic area; (iii) data resource; and (iv) scale of analysis. Their review examined the extent to which ML has been applied in several areas of earthquake engineering, including structural control for earthquake mitigation, seismic fragility assessment, system identification and damage detection, and seismic hazard analysis. Zhang et al. [9] presented a ML framework to assess post-earthquake structural safety. They proposed a systematic methodology for generating a reliable dataset for any damaged building, which included the incorporation of the concepts of response and damage patterns. The residual collapse capacity of the damaged structure was evaluated, using incremental dynamic analysis with sequential ground motions. ML algorithms were employed, to map response and damage patterns to the safety state of the building, based on a pre-determined threshold of residual collapse capacity.

Nguyen et al. [10] applied ML techniques to predict the seismic responses of 2D steel moment-resisting frames under earthquake motions. They used two popular ML algorithms—Artificial Neural Network (ANN) and eXtreme Gradient Boosting (XGBoost)—and took into consideration more than 22,000 non-linear dynamic analyses, on 36 steel moment frames of various structural characteristics, under 624 earthquake records with peak accelerations greater than 0.2 g. Both ML algorithms were able to reliably estimate the seismic drift responses of the structures, while XGBoost showed the best performance. Sadeghi Eshkevari et al. [11] proposed a physics-based recurrent ANN that was capable of estimating the dynamic response of linear and non-linear multiple degrees of freedom systems, given the ground motions. The model could estimate a broad set of responses, such as acceleration, velocity, displacement, and the internal forces of the system. The architecture of the recurrent block was inspired by differential equation solver algorithms. The study demonstrated that the network could effectively capture various non-linear behaviors of dynamic systems with a high level of accuracy, without requiring prior information or excessively large datasets.

In the interesting work of Abd-Elhamed et al. [12], logical analysis of data (LAD) was employed to predict the seismic responses of structures. The authors used real ground motions, considering a variation of earthquake characteristics, such as soil class, characteristic period, time step of records, peak ground displacement, peak ground velocity, and peak ground acceleration. The LAD model was compared to an ANN model, and was proven to be an efficient tool with which to learn, simulate, and predict the dynamic responses of structures under earthquake loading. Gharehbaghi et al. [13] used multi-gene genetic programming (MGGP) and ANNs to predict seismic damage spectra. They employed an inelastic SDOF system under a set of earthquake ground motion records, to compute exact spectral damage, using the Park–Ang damage index. The ANN model exhibited better overall performance, yet the MGGP-based mathematical model was also useful, as it managed to provide closed mathematical expressions for quantifying the potential seismic damage of structures.

Kazemi et al. [14] proposed a prediction model for seismic response and performance assessment of RC moment-resisting frames. They conducted incremental dynamic analyses (IDAs) of 165 RC frames with 2 to 12 stories and bay length ranging from 5.0 m to 7.6 m, ending up with a total of 92,400 data points for training the developed data-driven models. The examined output parameters were the maximum interstory drift ratio and the median of the IDA curves, which can be used to estimate the seismic limit state capacity and performance assessment of RC buildings. The methodology was tested in a five-story

RC building with very good results. Kazemi and Jankowski [15] used supervised ML algorithms in Python, to find median IDA curves for predicting the seismic limit-state capacities of steel moment-resisting frames considering soil–structure interaction effects. They used steel structures of two to nine stories subjected to three ground motion subsets as suggested by FEMA-P695, and 128,000 data points in total. They developed a user-friendly graphical user interface (GUI) to predict the spectral acceleration $S_a(T_1)$ of seismic limit-state performance levels using the developed prediction models. The developed GUI mitigates the need for computationally expensive, time-consuming, and complex analysis, while providing the median IDA curve including soil–structure interaction effects.

Wakjira et al. [16] presented a novel explainable ML-based predictive model for the lateral cyclic response of post-tensioned base rocking steel bridge piers. The authors implemented a wide variety of nine different ML techniques, ranging from the simple to most advanced ones, to generate the predictive models. The obtained results showed that the simplest models were inadequate to capture the relationship between the input factors and the response variables, while advanced models, such as the optimized XGBoost, exhibited the best performance with the lowest error. Simplified and approximate methods are particularly useful in engineering practice and have been successfully used by various researchers in structural dynamics and earthquake engineering related applications, such us the evaluation of the seismic performance of steel frames [17] and others.

The novelty of the present work consists of the development of new optimized ML models for the accurate and computationally efficient predictions of the fundamental eigenperiod, the maximum displacement as well as the base shear force of multi-story shear buildings. Four different ML algorithms are compared in terms of their prediction performance. The interpretation and explanation are elaborated using the permutations explainers of the SHAP methodology. In addition, a web application is developed based on the optimized ML models, to be easily used by engineers in practice. The remainder of the paper is organized as follows. Section 2 defines the problem formulation, followed by the description of the dataset and the exploratory data analysis in Section 3. Section 4 provides an overview of ML algorithms, followed by the ML pipelines and performance results of Section 5 and a discussion on interpretability of the results in Section 6. Section 7 presents and discussed the test case scenarios, while Section 8 presents the web application that has been developed and deployed for broad and open use. In the end, a short discussion and the conclusions of the study are presented.

2. Problem Formulation

The response spectrum modal analysis (RSMA) is a method to estimate the structural response to short, non-deterministic, and transient dynamic events. Examples of such events are earthquakes and shocks. Since the exact time history of the load is not known, it is difficult to perform a time-dependent analysis. The method requires the calculation of the natural mode shapes and frequencies of a structure during free vibration. It uses the mass and stiffness matrices of a structure to find the various periods at which it will naturally resonate, and it is based on mode superposition, i.e., a superposition of the responses of the structure for its various modes, and the use of a response spectrum. The idea is to provide an input that gives a limit to how much an eigenmode having a certain natural frequency and damping can be excited by an event of this type. The response spectrum is used to compute the maximum response in each mode, instead of solving the time history problem explicitly using a direct integration method. These maxima are non-concurrent and for this reason the maximum modal responses for each mode cannot be added algebraically. Instead, they are combined using statistical techniques, such as the square root of the sum of the squares (SRSS) method or the more complex and detailed complete quadratic combination (CQC) method. Although the response spectrum method is approximate, it is broadly applied in structural dynamics and is the basis for the popular equivalent lateral force (ELF) method. In the following subsections, a brief description of the RSMA

for multi-story structures is provided based on fundamental concepts of the single degree of freedom structural system.

2.1. Response Analysis of MDOF Systems

An idealized single degree of freedom (SDOF) shear building system has a mass m located at its top and stiffness k which is provided by a vertical column. For such a system without damping, the circular frequency ω, the cyclic frequency f and the natural period of vibration (or eigenperiod) T are given by the following formulas:

$$\omega = \sqrt{\frac{k}{m}} \qquad f = \frac{\omega}{2\pi} \qquad T = 2\pi\sqrt{\frac{m}{k}} \tag{1}$$

Similar to the SDOF system, a multi-story shear building, idealized as a multi-degree of freedom (MDOF) system is depicted in Figure 1, with the numbering of the stories from bottom to top. The vibrating system of the figure has n stories and n degrees of freedom (DOFs), denoted as the horizontal displacements u_i ($i = \{1, 2, \cdots, n\}$) at the top of each story. The dynamic equilibrium of a MDOF structure under earthquake excitation can be expressed with the following equation of motion at any time t:

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = -\mathbf{M} \cdot \mathbf{r} \cdot \ddot{u}_g(t) \tag{2}$$

where $\mathbf{M}(n \times n)$ is the mass matrix of the structure holding the masses m_i at its diagonal; $\mathbf{K}(n \times n)$ is the stiffness matrix; $\mathbf{C}(n \times n)$ represents the damping matrix, $\mathbf{r}(n \times n)$ is the influence coefficient vector; $\ddot{\mathbf{u}}(t)$, $\dot{\mathbf{u}}(t)$, $\mathbf{u}(t)$ (all $n \times 1$) are the acceleration, velocity, and displacement vectors, respectively, and $\ddot{u}_g(t)$ is the ground motion acceleration, applied to the DOFs of the structure defined by the vector $\mathbf{r}$.

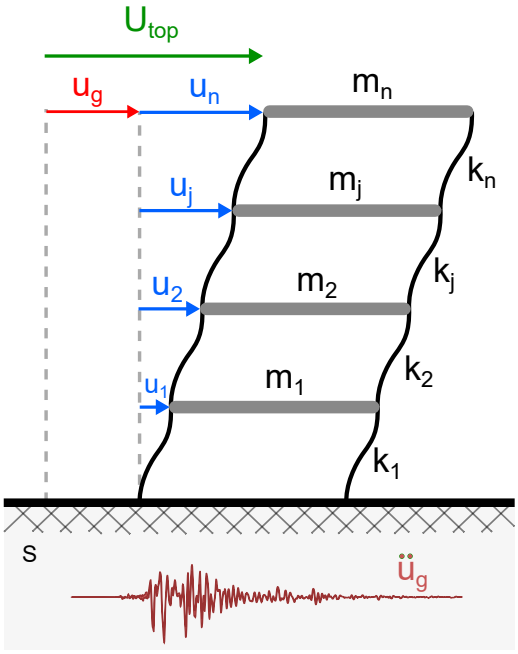


Figure 1. Multi-story shear building model with n DOFs.

The MDOF system has n natural frequencies ω_i ($i = 1, 2, \ldots, n$) which can be found from the characteristic equation:

$$\left|\mathbf{K} - \omega_i^2 \cdot \mathbf{M}\right| = 0 \tag{3}$$

By solving the determinant of Equation (3), one can find the eigenvalues λ_i of mode i which are the squares of the natural frequencies ω_i of the system ($\lambda_i = \omega_i^2$). Then,

the eigenvectors (or mode shapes or eigenmodes $\boldsymbol{\phi}_i$ (each $n \times 1$) can be found by the following equation:

$$\left(\mathbf{K} - \omega_i^2 \cdot \mathbf{M}\right) \cdot \boldsymbol{\phi}_i = \mathbf{0} \tag{4}$$

Equation (4) represents a generalized eigenvalue problem, which is a classic problem in mathematics. The solution of this problem involves a series of matrix decompositions which can be computationally expensive, especially for large systems with many DOFs.

Let the displacement response of the MDOF system be expressed as

$$\mathbf{u}(\mathbf{t}) = \boldsymbol{\Phi} \cdot \mathbf{y}(\mathbf{t}) \tag{5}$$

where $\mathbf{y}(\mathbf{t})$ represents the modal displacement vector and $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_n]$ is the matrix containing the eigenvectors. Substituting Equation (4) in Equation (2) and pre-multiply by $\boldsymbol{\Phi}^T$ we take

$$\underbrace{\boldsymbol{\Phi}^{\mathbf{T}}\mathbf{M}\boldsymbol{\Phi}}_{\mathbf{M}^*}\ddot{\mathbf{y}}(t) + \underbrace{\boldsymbol{\Phi}^{\mathbf{T}}\mathbf{C}\boldsymbol{\Phi}}_{\mathbf{C}^*}\dot{\mathbf{y}}(t) + \underbrace{\boldsymbol{\Phi}^{\mathbf{T}}\mathbf{K}\boldsymbol{\Phi}}_{\mathbf{K}^*}\mathbf{y}(t) = -\boldsymbol{\Phi}^{\mathbf{T}} \cdot \mathbf{M} \cdot \mathbf{r} \cdot \ddot{u}_g(t) \tag{6}$$

where $\mathbf{M}^*$, $\mathbf{C}^*$ and $\mathbf{K}^*$ are the generalized mass, generalized damping, and generalized stiffness matrices, respectively. By virtue of the properties of the matrix $\boldsymbol{\Phi}$, the matrices $\mathbf{M}^*$, $\mathbf{K}^*$, and $\mathbf{C}^*$ are all diagonal matrices and Equation (6) reduces to the following

$$\ddot{y}_i(t) + 2\xi_i \cdot \omega_i \cdot \dot{y}_i(t) + \omega_i^2 \cdot y_i(t) = -\Gamma_i \cdot \ddot{u}_g(t) \qquad i = \{1, 2, 3, \ldots, n\} \tag{7}$$

where $y_i(t)$ is the modal displacement response of the i^{th} mode, ξ_i is the modal damping ratio of the i^{th} mode and Γ_i is the modal participation factor for the i^{th} mode, expressed by

$$\Gamma_i = \frac{\boldsymbol{\phi}_i^T \mathbf{M}\mathbf{r}}{m_i^*} \tag{8}$$

where $m_i^* = \boldsymbol{\phi}_i^T \mathbf{M} \boldsymbol{\phi}_i$ is the i-th element of the diagonal matrix $\mathbf{M}^*$. Equation (7) represents n second order differential equations (i.e., similar to that of a SDOF system), the solution of which will provide the modal displacement response $y_i(t)$ for the i^{th} mode. Subsequently, the displacement response in each mode of the MDOF system can be obtained by Equation (5) using the $y_i(t)$.

2.2. Response Spectrum

In this work, we use the design spectrum for elastic analysis, as described in §3.2.2.5 of Eurocode 8 (EC8) [18]. The inelastic behavior of the structure is taken into account indirectly by introducing the behavior factor q. Based on this, an elastic analysis can be performed, with a response spectrum reduced with respect to the elastic one. The behavior factor q is an approximation of the ratio of the seismic forces that the structure would experience if its response was completely elastic with 5% viscous damping, to the seismic forces that may be used in the design, with a conventional elastic analysis model, still ensuring a satisfactory response of the structure. For the horizontal components of the seismic action, the design spectrum, $S_d(T)$, is defined as

$$S_d(T) = \begin{cases} a_g \cdot S \cdot \left[\frac{2}{3} + \frac{T}{T_B} \cdot \left(\frac{2.5}{q} - \frac{2}{3}\right)\right] & \text{, if} \quad 0 \le T \le T_B \\ a_g \cdot S \cdot \frac{2.5}{q} & \text{, if} \quad T_B \le T \le T_C \\ a_g \cdot S \cdot \frac{2.5}{q} \cdot \left[\frac{T_C}{T}\right] \ge \beta \cdot a_g & \text{, if} \quad T_C \le T \le T_D \\ a_g \cdot S \cdot \frac{2.5}{q} \cdot \left[\frac{T_C \cdot T_D}{T^2}\right] \ge \beta \cdot a_g & \text{, if} \quad T_D \le T \end{cases} \tag{9}$$

where T is the vibration period of a linear SDOF system, S is the soil factor, T_B and T_C are the lower and upper limits of the period of the constant spectral acceleration branch, respectively, T_D is the value defining the beginning of the constant displacement response

range of the spectrum, a_g is the design ground acceleration on type 'A' ground and β is the lower bound factor for the horizontal design spectrum, with a recommended value of 0.2. Although q introduces a non-linearity into the system, for the sake of simplicity, in this study we assume elastic behavior of the structure by taking q equal to 1. It has to be noted that we do not use the horizontal elastic response spectrum which is described in §3.2.2.2 of EC8, but rather the design spectrum for elastic analysis of §3.2.2.5, for the case $q = 1$. The two are almost the same, but there are also some minor differences.

2.3. Response Spectrum Method for MDOF Systems

Given the spectrum, Equation (7) is forming the equation of motion of a SDOF system. The maximum modal displacement response $y_{i,\max}$ is found from the response spectrum as follows:

$$y_{i,\max} = |y_i(t)|_{\max} = \Gamma_i \cdot \frac{S_d(T_i)}{\omega_i^2} \tag{10}$$

Consequently, the maximum displacement ($u_{i,\max}$) and acceleration ($\ddot{u}_{i,\max}$) response of the MDOF system in the i^{th} mode are given as follows:

$$\begin{aligned} u_{i,\max} &= \phi_i \cdot y_{i,\max} \\ \ddot{u}_{i,\max} &= \phi_i \cdot \Gamma_i \cdot S_d(T_i) = \phi_i \cdot \omega_i^2 \cdot y_{i,\max} = \phi_i \cdot u_{i,\max} \end{aligned} \tag{11}$$

In each mode of vibration, the required response quantity of interest Q, i.e., displacement, shear force, bending moment, etc., of the MDOF system can be obtained using the maximum response obtained by Equation (11). However, the final maximum response Q_{max}, is obtained by combining the response in each mode using a modal combination rule. In this study, the commonly square root of sum of squares (SRSS) rule is used as follows:

$$Q_{\max} = \sqrt{\sum_{i=1}^{n} Q_i^2} \tag{12}$$

The SRSS method of combining maximum modal responses is fundamentally sound when the modal frequencies are well separated.

3. Dataset Description and Exploratory Data Analysis

The dataset was generated from 1995 results of dynamic response analyses of multi-story shear buildings of various configurations using the response spectrum method described in Section 2. More specifically, the dataset consists of 3 features, namely (i) <Stories>, the number of stories in the shear building; (ii) <$\tilde{k}$>, the normalized stiffness over the mass of each story; and (iii) <Ground Type>, the ground type as the code provision (EC8) dictates. In addition, the dataset is completed with 3 targets, namely (i) <T_1>, the fundamental eigenperiod of the building; (ii) <U_{top}>, the horizontal displacement at the top story; and (iii) <$\tilde{V}_b$>, the normalized base shear force over the mass of each story of the building.

3.1. Dataset Description

In this study, we assume a constant k and m for all stories of the building, i.e., k_i and m_i remain constant for each story i. There is no change in the mass or stiffness of each story, along the height of the building. For such buildings, the response of the structure is characterized by the ratio k/m rather than the individual values of k and m and this is the reason why k/m, denoted as <$\tilde{k}$> (normalized stiffness over mass), is taken as the input in the analysis, instead of taking into account the individual k and m for each story. The unit used for $\tilde{k}$ is (N/m)/kg which is equivalent to s^{-2}. The normalized stiffness ranges from 2000 to 12,000 s^{-2} (with a step of 500, i.e., 21 unique values), while the number of stories ranges from 2 to 20 (with a step of 1, resulting in 19 values), covering a wide range of the structures and representing the majority of typical multi-story shear buildings that can be found in practice.

The normalized base shear force over the mass of each story of the building has unit N/kg, which is equivalent to $m \cdot s^{-2}$. The ground acceleration a_g for this study is kept constant at 1 g = 9.81 m/s^2 as it affects the results in a linear way, since we assume elastic behavior ($q = 1$). As a result, all outputs are calculated with reference to an acceleration of 1 g. If another value is used for the ground acceleration, as is performed in the examined test scenarios, then the outputs of the model need to be multiplied with this ground acceleration value to obtain the correct results. A damping ratio of 5% was considered in all analyses.

All the targets, along with the input parameter <$\tilde{k}$> are treated as continuous variables, while the remaining features are treated as integer variables. For the <Ground Type> feature, which natively takes values from the list of ['A', 'B', 'C', 'D', 'E'], the ordinal encoding was used. In this encoding, each category value is assigned to an integer value due to the natural ordered relationship between each other, i.e., a type 'B' ground, is "worse" than a type 'A' ground, etc. Hence, the machine learning algorithms are able to understand and harness this relationship.

The final dataset, consists of 1995 observations in total, which is the product of 19 × 21 × 5, where 19 is the different numbers of stories, 21 are the different values of the normalized stiffness over the mass of each story, and 5 are the different ground types considered.

3.2. Exploratory Data Analysis

Understanding the data is very important before building any machine learning model. The statistical parameters and the distributions of dataset's variables provide useful insights on the dataset and presented in Table 1 and Figure 2, respectively. From the latter one, it can be observed that all targets follow a right skewed unimodal distribution with platykurtic kurtosis (flatter than the normal distribution).

Figure 3 depicts the box and Whisker plots for features (orange) and targets (moonstone blue). The red vertical line shows the median of each distribution. The box shows the interquantile range (IQR) which measures the spread of the middle half of the data and contains 50% of the samples, defined as $IQR = Q3 - Q1$, where $Q1$ and $Q3$ are the lower and upper quartiles, respectively. The black horizontal line shows the interval from the lower outlier gate $(Q1 - 1.5 \cdot IQR)$ to the upper outlier gate $(Q3 + 1.5 \cdot IQR)$. As a result, the blue dots represent the "outliers" in each target, according to interquantile range (IQR) method. Often outliers are discarded because of their effect on the total distribution and statistical analysis of the dataset. However, in this situation, the occasional 'extreme' building configurations (i.e., very flexible structures) cause an outlier that is outside the usual distribution of the dataset but is still a valid measurement.

Table 1. Statistical parameters of the dataset.

	Stories	$\tilde{k}$	**Ground Type**	T_1	U_{top}	$\tilde{V}_b$
Unit		**[s^{-2}]**		**[s]**	**[m]**	**[$m \cdot s^{-2}$]**
count	1995	1995	1995	1995	1995	1995
mean	11.000	7000.000	2.000	0.604	0.296	0.213
std	5.479	3028.600	1.414	0.341	0.240	0.101
skew	0.000	0.000	0.000	0.782	1.265	0.538
kurtosis	−1.207	−1.205	−1.300	0.435	2.162	0.072
min	2.000	2000.000	0.000	0.093	0.004	0.032
25%	6.000	4500.000	1.000	0.330	0.104	0.142
50%	11.000	7000.000	2.000	0.567	0.254	0.201
75%	16.000	9500.000	3.000	0.801	0.416	0.282
max	20.000	12,000.000	4.000	1.834	1.571	0.553
Type	[int]	[float]	[int]	[float]	[float]	[float]

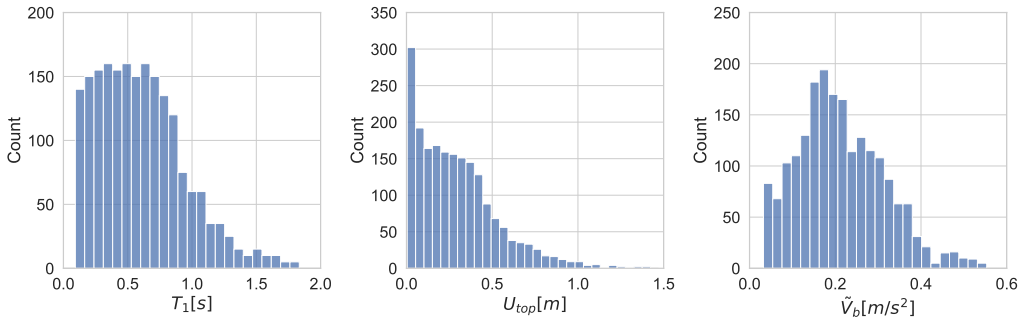


Figure 2. Histograms of the targets T_1, U_{top}, and $\tilde{V}_b$.

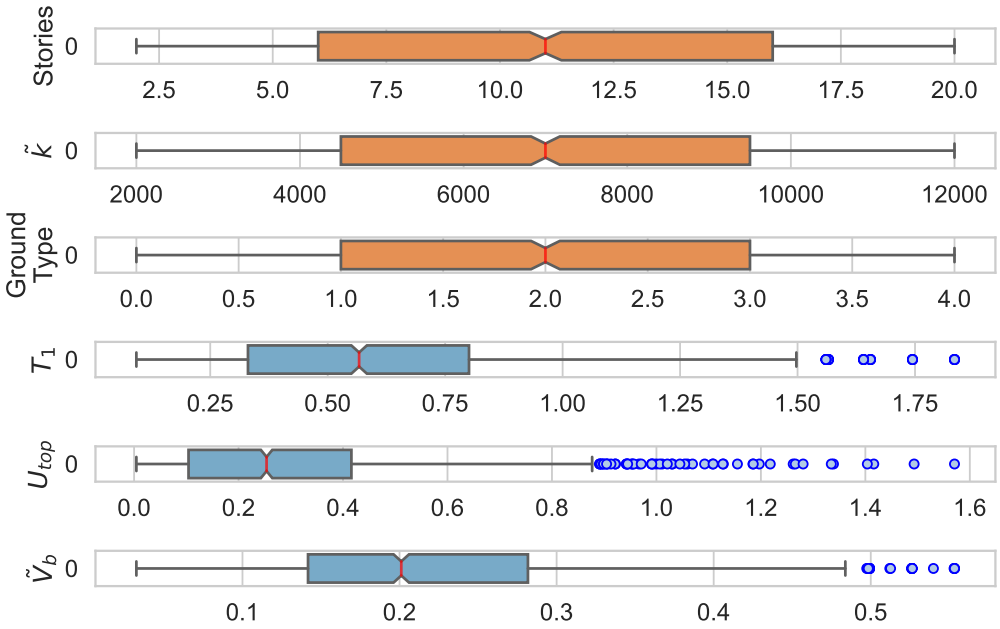


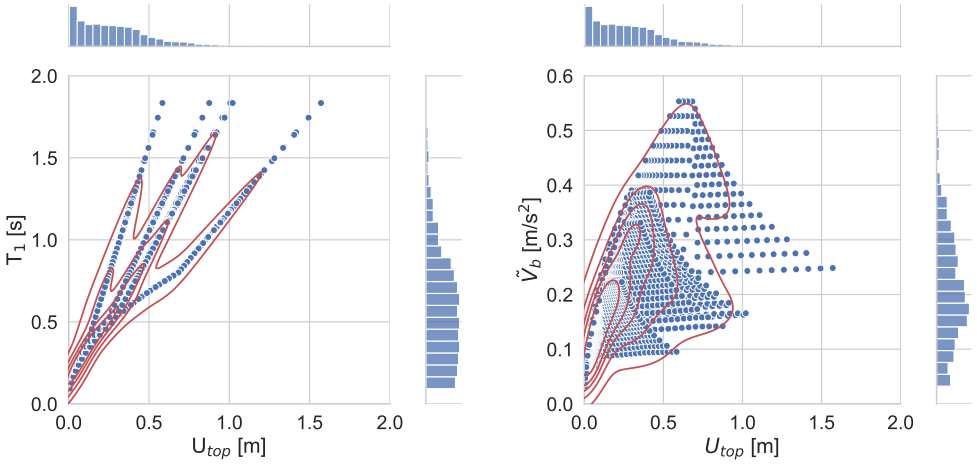
Figure 3. Box and Whisker plots for features (orange) and targets (moonstone blue). Red vertical line shows the median of each distribution. The blue dots represent the outliers in the distribution, according to IQR method.

In Figure 4, the joint plots with their kernel density estimate (KDE) plots for u_{top} feature against T_1 and $\tilde{V}_b$ are also depicted. KDE is a method for visualizing the distribution of observations in a dataset, analogous to a histogram and represents the data using a continuous probability density curve in two dimensions. Unlike a histogram, a KDE plot smooths the observations with a Gaussian kernel, producing a continuous density estimate. It can be observed that T_1 and $\tilde{V}_b$ are correlated with a 'linear' type relation. This relation can be also derived from the high correlation values depicted in Figure 5 that shows the correlation matrix of the dataset features and targets, including also the Pearson product-moment correlation coefficient. The Pearson product-moment correlation coefficient (ρ) is used to measure the correlation intensity between a pair of independent random variables (x, y), according to the following relation

$$\rho(x,y) = \frac{COV(x,y)}{\sigma_x \sigma_y} \tag{13}$$

where COV is the covariance between the two random variables (x, y) and σ_x, σ_y is the standard deviation of x, y, respectively. $|\rho| > 0.8$ represents a *strong* relationship between x and y, values between 0.3 and 0.8 represent *medium* relationship, while $|\rho| < 0.3$ represents

a *weak* relationship. It is shown that the number of stories, has a strong relationship with T_1 and U_{top}, while Ground Type has no relationship with the number of stories and $\tilde{k}$.



(**a**) (**b**)

Figure 4. Joint and KDE plots of U_{top} vs. (**a**) T_1 and (**b**) $\tilde{V}_b$.

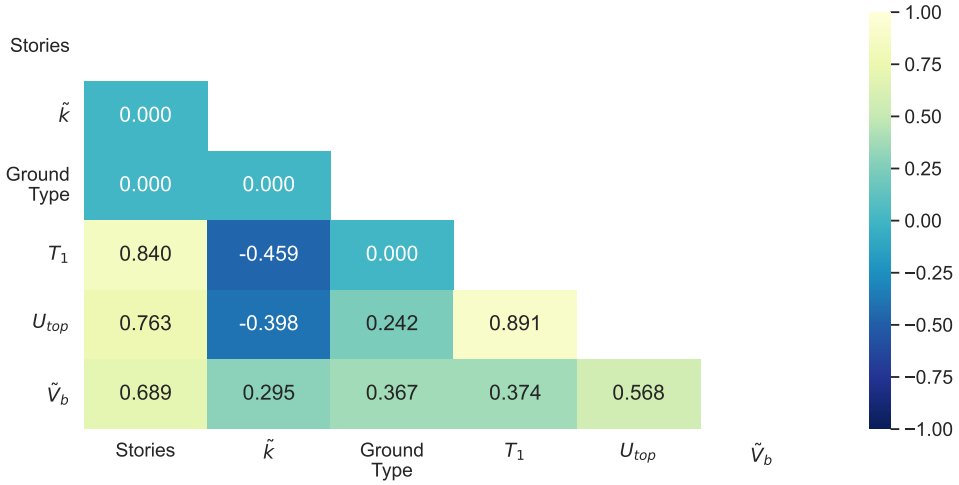


Figure 5. Features and targets correlation matrix.

4. Overview of ML Algorithms

This study estimates the dynamic behavior of shear multi-story buildings in terms of predicting the fundamental eigenperiod (T_1), the roof top displacement (u_{top}), and the normalized base shear ($\tilde{V}_b$) by using four ML algorithms including Ridge Regressor (RR), Random Forest (RF) regressor, Gradient Boosting (GB), and Category Boosting (CB) regressor. All considered algorithms (except RR) belong to ensemble methods which seek better predictive performance by combining the predictions from multiple models usually in the form of decision trees by means of the bagging (bootstrap aggregating) and boosting ensemble learning techniques. Bagging involves fitting many decision trees (DTs) on different samples of the same dataset and averaging the predictions, while, in boosting, the ensemble members are added sequentially by correcting the predictions made by preceding models and the method outputs a weighted average of the predictions. Ensemble learning techniques eliminate any variance, thereby reducing the overfitting of models. In the following sections, an overview of each ML algorithm is provided, along with its strong and weak points.

4.1. Ridge Regression (RR)

With the absence of constraints, every model in machine learning will overfit the data and make unnecessary complex relationships. To avoid this, the regularization of data is needed. Regularization simplifies excessively complex models that are prone to be overfit and can be used to any machine learning model. Ridge regression [19] is a regularized version of linear regression that uses the mean squared error loss function (LF) and applies L2 Regularization. In L2 Regularization (also known as Tikhonov Regularization), the penalty term is applied into the square of weights (w) to the loss function as follows:

$$\text{Regularization}(L2) = \text{LF} + \lambda \sum_{i=1}^{m} w_i^2 \tag{14}$$

Consequently, the cost function $J(\theta)$ in Ridge Regression takes the following form

$$J(\theta) = \underbrace{\frac{1}{m} \sum_{i=1}^{m} \left(y_i - \hat{y}_i \right)^2}_{\text{Loss Function}} + \underbrace{\lambda \sum_{j=1}^{n} w_j^2}_{\text{Penalty}} \tag{15}$$

where m is the total number of observations in the dataset, n is the number of features in the dataset, y and $\hat{y}$ are the ground truth and the predicted values of the regression model, respectively, and λ is the penalty term which express the strength of regularization. The penalization in the sum of the squared weights reduces the variance of the estimates and the model, i.e., it shrinks the weights and, thus, reduces the standard errors. The penalty term serves to reduce the magnitude of the weights, and it also helps to prevent overfitting. As a result, RR can provide improved predictive accuracy and stability.

Ridge regression also has the ability to handle non-linear relationships between predictor and outcome variables, in contrast to linear regression. It is more robust to collinearity than linear regression and it can be applied to small datasets, while no perfect normalization of data is required. However, RR can be computationally expensive if the dataset is large. In addition, its results are difficult to interpret because the L2 regularization term modifies the weights. This is because the cost function contains a quadratic term, which makes it more difficult to optimize. In addition, RR only provides a closed-form approximation of the solution and can produce unstable results if outliers are present in the dataset.

Although, we *a priori* know that Ridge Regression will not able to compete with the other ensemble models, it is still selected as a simplistic method for a rough approximation of the model to be fitted.

4.2. Random Forest Regressor (RF)

Decision trees are simple tree-like models of decisions that work well for many problems, but they can also be unstable and prone to overfitting. The Random Forest developed by Breiman [20] overcomes these limitations by using an ensemble of decision trees as the weak learners, where each tree is trained on a random subset of the data and features (hence the name "Random Forest"). The subsets of the training data are created by random sampling with replacement (bootstrap sampling), thus, some data points may be included in multiple subsets, while others may not be included at all. Each model in the ensemble is trained independently using the same learning algorithm and hyperparameters, but with its own subset of the training data. The predictions from each tree are then combined by taking the average (Figure 6). Therefore, this randomness helps reduce the variance of the model and the risk of overfitting problems in the decision tree method.

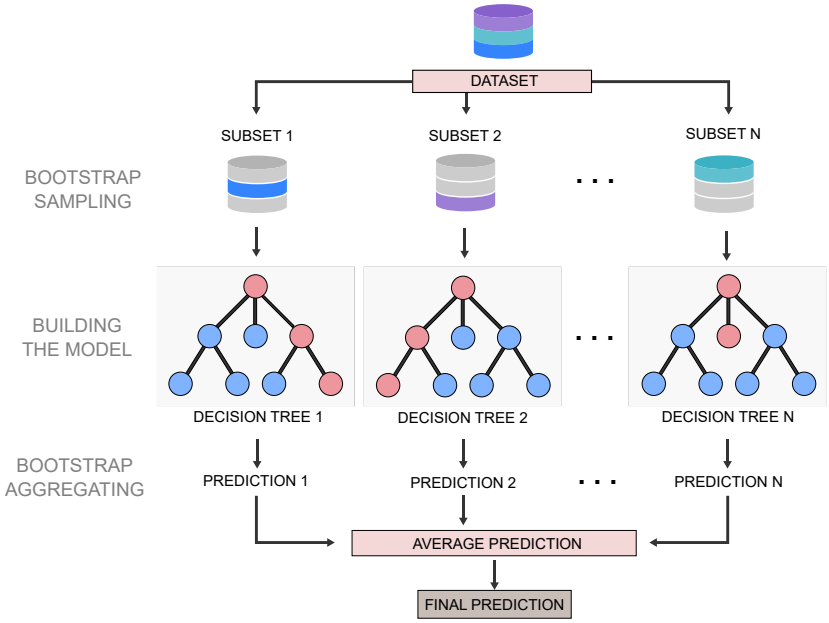


Figure 6. Random Forest algorithm flowchart.

Random Forest is one of the most accurate machine learning algorithms which inherits the merits of the decision tree algorithm. It can work well with both categorical and continuous variables and can handle large datasets with thousands of features. Random Forest is a robust algorithm that can deal with noisy data and outliers and can generalize well to unseen data without the need of normalization as it uses a rule-based approach. Despite being a complex algorithm, it is fast and provides a measure of feature importance, which can help in feature selection and data understanding.

Although RF is less prone to overfitting than a single decision tree, it can still overfit the data if the number of trees in the forest is too high or if the trees are too deep. Random Forest can be less interpretable than a single decision tree because it involves multiple trees. Thus, it can be difficult to understand how the algorithm arrived at a particular prediction. The training time of RF can be longer compared to other algorithms, especially if the number of trees and their depth are high. Random Forest requires more memory than other algorithms because it stores multiple trees. This can be a problem if the dataset is large. Overall, RF is a handy and powerful algorithm where its default parameters are often good enough to produce acceptable results.

4.3. Gradient Boosting Regressor (GB)

Gradient Boosting is one of the variants of ensemble methods in which multiple weak models (decision trees) are combined to obtain better performance as a whole. Gradient Boosting algorithm was developed by Friedman [21] and uses decision trees as weak learners. In general, weak learners are not necessary to have the same structure, so they can capture different outputs from the data. In Gradient Boosting, the loss function of each weak learner is minimized using the gradient descent procedure, a global optimisation algorithm which can apply to any loss function that is differentiable. As shown in Figure 7, the residual (loss error) of the previous tree is taken into account in the training of the following tree. By combining all trees, the final model is able to capture the residual loss from the weak learners.

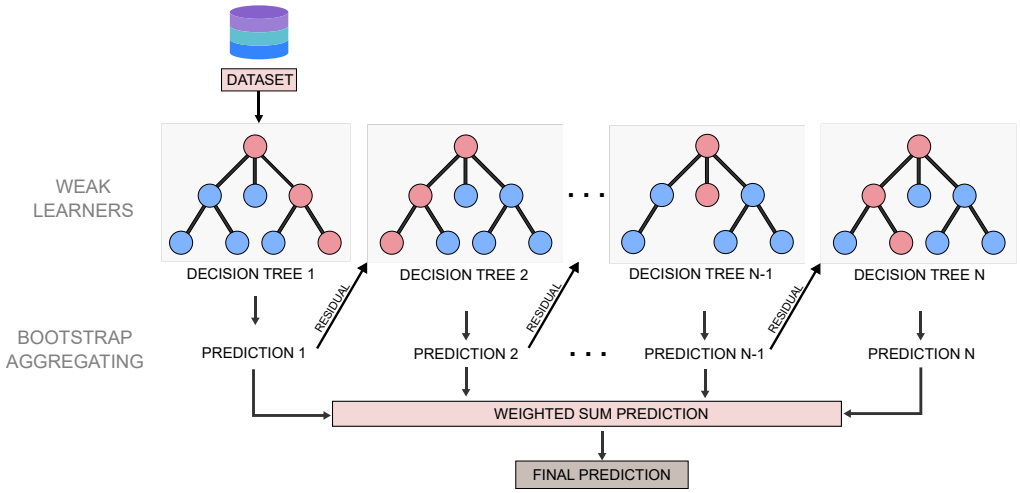


Figure 7. Gradient Boosting algorithm flowchart.

To better understand how Gradient Boosting works, we present below the steps involved.

Step 1. Create a base tree with single root node that acts as the initial guess for all samples.
Step 2. Create a new tree from the residual (loss errors) of the previous tree. The new tree in the sequence is fitted to the negative gradient of the loss function with respect to the current predictions.
Step 3. Determine the optimal weight of the new tree by minimizing the overall loss function. This weight determines the contribution of the new tree in the final model.
Step 4. Scale the tree by learning rate that determines the contribution of the tree in the prediction.
Step 5. Combine the new tree with all the previous trees to predict the result and repeat Step 2 until a convergence criterion is satisfied (number of trees exceeds the maximum limit achieved or the new trees do not improve the prediction).

The final prediction model is the weighted sum of the predictions of all the trees involved in the previous procedure, with better-performing trees having a higher weight in the sequence.

In Gradient Boosting, every tree is built one at a time, whereas Random Forests build each tree independently. Thus, the Gradient Boosting algorithm runs in a fixed order, and that sequence cannot change, leading to only sequential evaluation. The Gradient Boosting algorithm is not known for being easy to read or interpret compared to other ensemble algorithms like Random Forest. The combination of trees in Gradient Boosting can be more complex and harder to interpret, although recent developments can improve the interpretability of such complex models. Gradient Boosting is sensitive to outliers since every estimator is obliged to fix the errors in the predecessors. Furthermore, the fact that every estimator bases its correctness on the previous predictors, makes the procedure difficult to scale up.

Overall, Gradient Boosting can be more accurate (under conditions depending on the nature of the problem and the dataset) than Random Forest, due to the sequential nature of the training process of trees which correct each other's errors. This attribute is capable of capturing complex patterns in the dataset, but it can still be prone to overfitting in noisy datasets.

4.4. CatBoost Regressor (CB)

CatBoost is a relatively new open-source machine learning algorithm which is based on Gradient Boosted decision trees. CatBoost was developed by Yandex engineers [22] and it focuses on categorical variables without requiring any data conversion in the preprocessing. CatBoost builds symmetric trees (each split is on the same attribute), unlike the Gradient Boosting algorithm, by using permutation techniques. This means that in every

split, leaves from the previous tree are split using the same condition. The feature-split pair that accounts for the lowest loss is selected and used for all the level's nodes. The balanced tree architecture decreases the prediction time while controlling overfitting as the structure serves as regularization. CB uses the concept of ordered boosting, a permutation-driven approach to train the model on a subset of data while calculating residuals on another subset. This technique prevents overfitting and the well-known *dataset shift*, a challenging situation where the joint distribution of features and targets differs between the training and test phases.

CatBoost supports all kinds of features, such as numeric, categorical, or text, which reduces the time of the dataset preprocessing phase. It is powerful enough to find any non-linear relationship between the model target and features and has great usability that can deal with missing values, outliers, and high cardinality categorical values on features without any special treatment. Overall, CatBoost is a powerful Gradient Boosting framework that can handle categorical features, missing values, and overfitting issues. It is fast, scalable, and provides good interpretability.

5. ML Pipelines and Performance Results

The ML models developed in this study are based on the dataset described in Section 3 and make use of the following open-source Python libraries, scikit-learn (RR, RF, GB) [23] and CatBoost (CB) [22]. Three different ML models are considered for predicting the fundamental eigenperiod (T_1), the horizontal displacement at the top story (U_{top}), and the normalized shear base over the mass of each story ($\tilde{V}_b$) of a shear building. The features of all models are the number of stories (Stories), the normalized stiffness over the mass of each story ($\tilde{k}$) and the Ground Type.

5.1. Cross Validation and Hyperparameter Tuning

The dataset is split into training and testing set, with 80% and 20% of the samples, respectively. The training set was validated via the k-fold cross-validation method as follows. Data are shuffled and divided into k equal sized subsamples. One of the k subsamples is used as a test (validation) set and the remaining $(k-1)$ subsamples are put together to be used as training data. Then a model is fitted using training data and evaluated using the test set. The process is repeated k times until each group has served as the validation set. The k results from each model are averaged to obtain the final estimation.

The advantage of the k-fold cross-validation method is that the bias and variance are significantly reduced, while the robustness of the model is increased. The testing set, with data that remain unseen by the models during the training, is used for the final test of the model performance and generalization. With the term *generalization* we refer to the model's ability to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model. The value of k depends on the size of the dataset in a way which does not increase the computational cost. In this study, the k value is set equal to 10.

Cross validation is performed together with the hyperparameter tuning in the data pipeline. Hyperparameter tuning is the process of selecting the optimized values for a model's parameters that maximize its accuracy. The optimal values of the hyperparameters for each model are found using extensive grid search, in which every possible combination of hyperparameters is examined to find the best model. The optimized values of the hyperparameters, along with the range of each ML model and algorithm, are presented in Table 2. The hyperparameter names correspond to those in the utilized Python libraries [23]. The hyperparameters not shown had been assigned the default values.

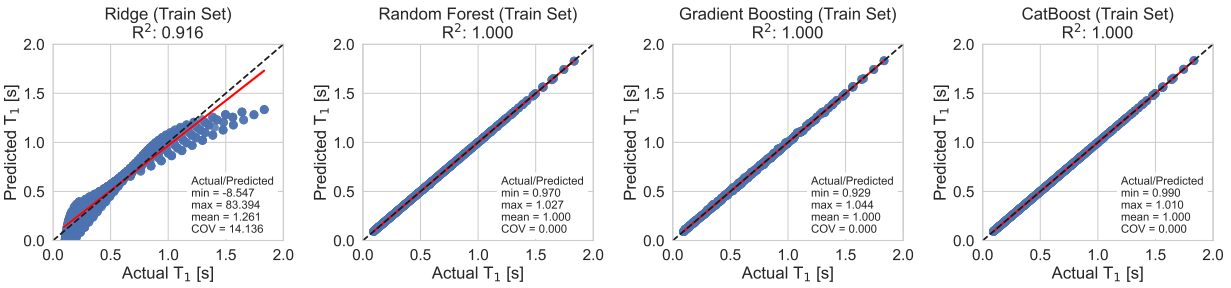
Table 2. Optimal hyper-parameters values for each ML algorithm and model found via grid search.

Algorithm	Hyper-Parameter	Search Range	Model Optimal Value		
			T_1	U_{top}	$\tilde{V}_b$
Ridge	alpha	[0, 0.1, 0.5, 1, 5, 10]	10	10	10
	max_iter	[50, 100, 500, 1000]	50	50	50
	solver	['svd', 'cholesky', 'lsqr']	lsqr	svd	svd
	tol	[0.0001, 0.001]	0.001	0.001	0.001
Random Forest	n_estimators	[10, 20, 50, 100, 500]	500	500	500
	max_depth	[2, 5, 10]	10	10	10
	criterion	['sqr', 'abs', 'fried', 'pois']	fried	fried	fried
	min_samples_split	[1, 2, 5, 10, 20]	5	5	5
	min_samples_leaf	[1, 2, 5]	1	1	2
	min_impurity_decrease	[0.01, 0.02, 0.05, 0.1, 0.2]	0.01	0.01	0.01
Gradient Boosting	n_estimators	[10, 20, 100, 500]	500	500	500
	learning_rate	[0.01, 0.1]	0.1	0.1	0.1
	criterion	['sqr', 'fried']	sqr	sqr	sqr
	min_samples_leaf	[1, 2, 5, 10]	1	1	10
	min_samples_split	[5, 10, 20, 100]	10	5	5
	max_depth	[1, 2, 5, 10]	10	5	10
CatBoost	n_estimators	[10, 20, 100, 500]	500	500	500
	learning_rate	[0.01, 0.1]	0.1	0.1	0.1
	l2_leaf_reg	[1, 2, 5, 10]	1	1	2
	bagging_temperature	[0.0, 0.1, 0.2, 0.5, 1.0]	0	0	0
	depth	[1, 2, 5, 10]	5	5	10

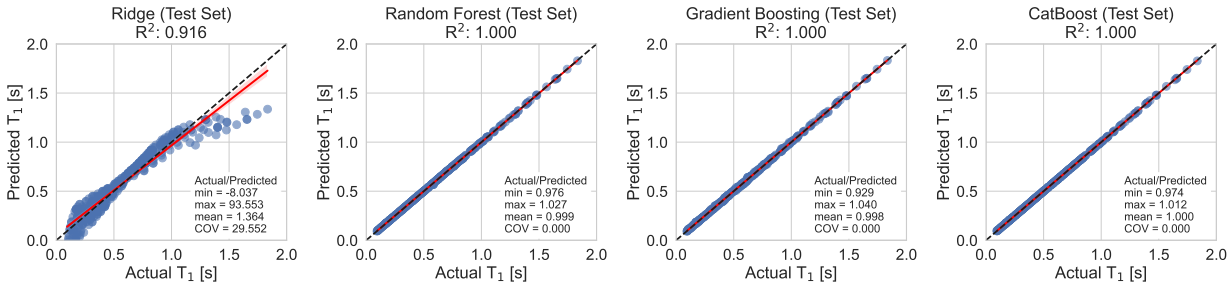
where "alpha": the constant that multiplies the L2 term, controlling regularization strength. | "max_iter": the maximum number of iterations for conjugate gradient solver. | "solver": the solver to use in the computational routines. | "tol": the precision of the solution (tol has no effect for solvers 'svd' and 'cholesky'). | "n_estimators": the number of trees in the forest. | "max_depth": the number of trees in the forest. | "criterion": the function to measure the quality of a split. Possible values are: 'sqr', 'abs', 'fried', and 'pois' which stand for 'squared_error', 'absolute_error', 'friedman_mse', and 'poisson', respectively. | "min_samples_split": the number of trees in the forest. | "min_samples_leaf": the minimum number of samples required to be at a leaf node. | "min_impurity_decrease": the value in which a node will be split, if this split induces, a decrease in the impurity greater than this. | "learning_rate": factor that shrinks the contribution of each tree. | "l2_leaf_reg": the coefficient of L2 regularization term of the cost function. | "bagging_temperature": parameter to define the settings of the Bayesian bootstrap and assign random weights to objects. The weights are sampled from exponential distribution if the value of this parameter is set to "1". All weights are equal to 1 if the value of this parameter is set to "0". Possible values are in the range [0;inf). The higher the value the more aggressive the bagging is.

Table 3 collects statistics of the fit time and test score for each model during the cross-validation and hyperparameter tuning process, which is performed on the same hardware configuration. It is shown that Ridge Regression has the lowest fit time for all the models (up to 395 speed-up when compared to the slowest), while CatBoost algorithm outperforms all the others in terms of scoring and exhibits the lowest standard deviation value.

Figures 8–10 show the performance of the ML models for predicting the T_1, U_{top} and $\tilde{V}_b$ of the shear buildings in the train and test datasets for the optimized hyperparameter values, accordingly. In general, the ensemble methods achieved higher accuracy compared to the Ridge Regression algorithm. However, the Ridge Regression algorithm managed to achieve acceptable results in the case of the T_1 model.

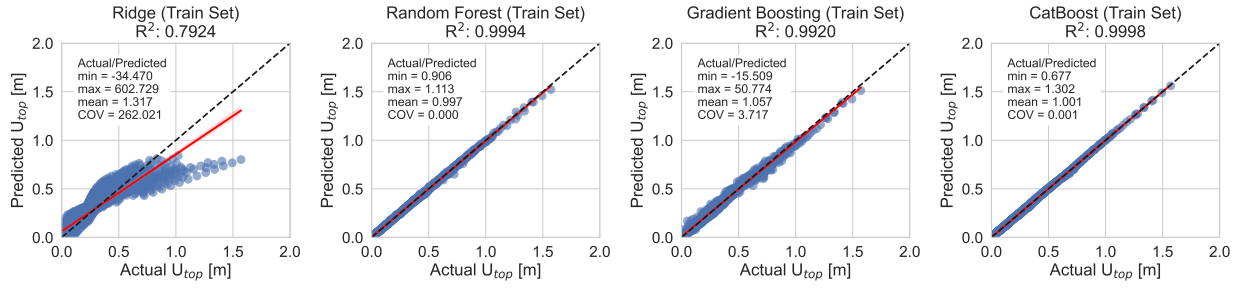


(**a**) Train set

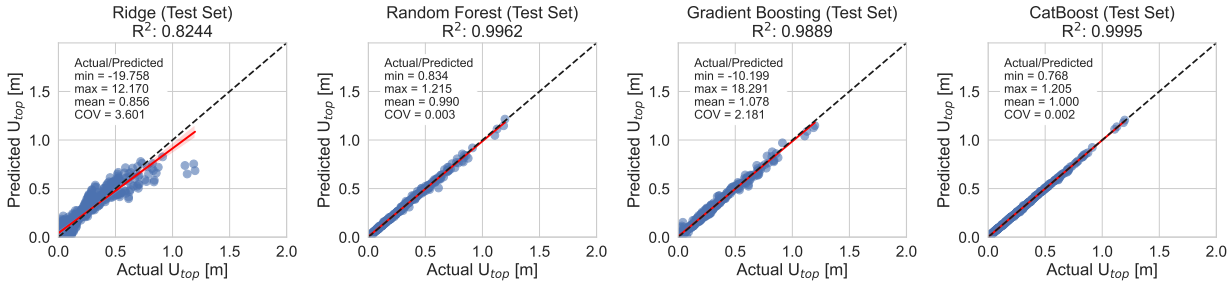


(**b**) Test set

Figure 8. Actual vs. Predicted plots for both (**a**) train and (**b**) test dataset for T_1 model.

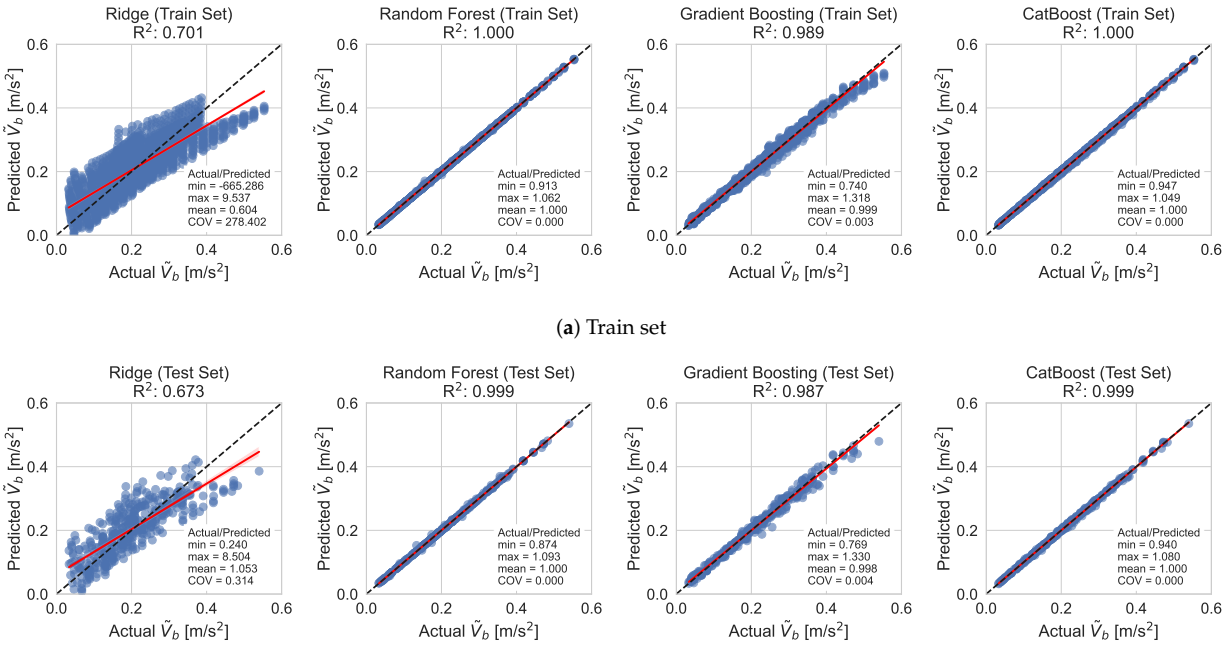


(**a**) Train set



(**b**) Test set

Figure 9. Actual vs. Predicted plots for both (**a**) train and (**b**) test dataset for U_{top} model.



(**a**) Train set

(**b**) Test set

Figure 10. Actual vs. Predicted plots for both (**a**) train and (**b**) test dataset for $\tilde{V}_b$ model.

Table 3. Cross-Validation performance for each ML algorithm and model.

Algorithm	Candidates	Model	Fit Time [s]		Test Score	
			Mean	Std Dev	Mean	Std Dev
Ridge	144	T_1	0.003	0.000	0.915	0.006
		U_{top}	0.003	0.000	0.800	0.028
		$\tilde{V}_b$	0.003	0.000	0.684	0.034
Random Forest	2880	T_1	1.184	0.073	0.999	0.001
		U_{top}	1.137	0.013	0.988	0.001
		$\tilde{V}_b$	1.072	0.001	0.990	0.002
Gradient Boosting	1024	T_1	0.919	0.011	1.000	0.000
		U_{top}	0.583	0.001	0.999	0.001
		$\tilde{V}_b$	0.839	0.006	0.999	0.000
CatBoost	640	T_1	0.627	0.087	1.000	0.000
		U_{top}	0.218	0.032	0.999	0.000
		$\tilde{V}_b$	0.508	0.063	0.999	0.000

5.2. Model Evaluation Metrics

To quantify the performance of the ML models, the well-known metrics *RMSE*, *MAE*, *MAPE*, and R^2 are used [24]. The definition of each metric is as follows

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - \hat{x}_i)^2} \tag{16}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{m}(x_i - \hat{x}_i) \tag{17}$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{m}\left|\frac{x_i - \hat{x}_i}{x_i}\right| \quad (18)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(x_i - \hat{x}_i)^2}{\sum_{i=1}^{m}(x_i - \hat{x})^2} \quad (19)$$

where m is the size of the dataset, x_i and $\hat{x}_i$ are the actual and predicted feature value for observation i, respectively.

The performance metrics of each ML algorithm and model are provided in Table 4. CatBoost and Random Forest were the two best-performing ML algorithms. CatBoost performed best for the fundamental eigenperiod T_1, and the horizontal displacement at the top U_{top}, with MAE values of 0.008 and 0.0034, respectively, for the test set, compared to 0.0013 and 0.0084 of the Random Forest. On the other hand, Random Forest had the best performance for the normalized base shear force over the mass of each story $\tilde{V}_b$, with MAE value of 0.0010 for the test set, compared to 0.0020 of the CatBoost algorithm. The other two algorithms, Ridge and Gradient Boosting showed larger values of MAE as well as worse values for the other metrics.

Table 4. Performance metrics of each ML algorithm and model. The finally selected algorithm (CatBoost) is highlighted with brown color.

ML Algorithm	RMSE		MAE		MAPE		R^2	
	Train	Test	Train	Test	Train	Test	Train	Test
				T_1				
Ridge	0.0098	0.0098	0.0743	0.0722	0.1916	0.1932	0.9163	0.9159
Random Forest	0.0000	0.0000	0.0006	0.0013	0.0010	0.0023	1.0000	1.0000
Gradient Boosting	0.0000	0.0000	0.0040	0.0041	0.0086	0.0097	0.9997	0.9997
CatBoost	0.0000	0.0000	0.0006	0.0008	0.0013	0.0020	1.0000	1.0000
				U_{top}				
Ridge	0.0123	0.0090	0.0754	0.0699	1.0630	1.0817	0.7924	0.8244
Random Forest	0.0000	0.0002	0.0035	0.0084	0.0150	0.0400	0.9994	0.9962
Gradient Boosting	0.0005	0.0006	0.0146	0.0164	0.1156	0.1488	0.9920	0.9889
CatBoost	0.0000	0.0000	0.0026	0.0034	0.0182	0.0238	0.9998	0.9995
				$\tilde{V}_b$				
Ridge	0.0031	0.0032	0.0440	0.0450	0.2833	0.3061	0.7010	0.6727
Random Forest	0.0000	0.0000	0.0004	0.0010	0.0024	0.0060	0.9999	0.9994
Gradient Boosting	0.0001	0.0001	0.0073	0.0080	0.0372	0.0441	0.9894	0.9866
CatBoost	0.0000	0.0000	0.0015	0.0020	0.0079	0.0110	0.9995	0.9990

In general, CatBoost comes first in accuracy with acceptable fit and predicted times for most of the cases, while Ridge Regression takes the trophy for being the fastest to fit the data. Overall, CatBoost appears to be the best model to move forward with, as it came first for, arguably, the most important metrics, although for the case of $\tilde{V}_b$ model, the Random Forest algorithm exhibited slightly better performance.

6. ML Interpretability

Machine learning models are often treated as "black boxes" which makes their interpretation difficult. To understand the main features that affect the prediction of a model, explainable machine learning techniques can be used to demystify their properties. Toward this, many explainability techniques have been developed. One which has gained increasing interest is the SHAP (SHapley Additive exPlanations) method introduced by Lundberg and Lee [25]. The method explains individual predictions and can be used for the quantification of relative feature importance. The SHAP method is based on the game

theoretically optimal Shapley values which measure the contribution to the outcome from each feature separately among all the input features.

6.1. Feature Importance

SHAP feature importance is an abstract approach to explain the predictions of a machine learning model. It provides an intuitive way to understand which features are most important to the prediction based on the magnitude of feature attributions, where large absolute Shapley values are the most important. The SHAP feature importance (FI) can be quantified using the following formula

$$FI_j = \frac{1}{n} \sum_{i=1}^{m} |s_j^{(i)}| \tag{20}$$

where m is the number of observations in the dataset and $s_j^{(i)}$ is the SHAP value of the feature j for observation i. Figure 11 shows the SHAP feature importance by decreasing importance for the best performing ML model. We see that the number of stories is the most important feature affecting all targets. On the other hand, the Ground Type feature, has no impact on the fundamental period T_1 of the structure, which is meaningful and it is expected according to the theory.

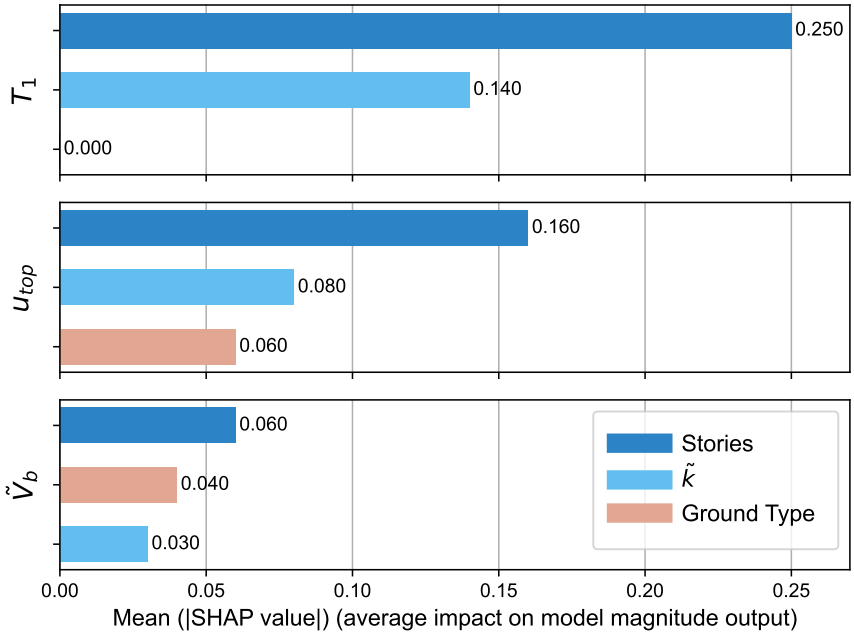
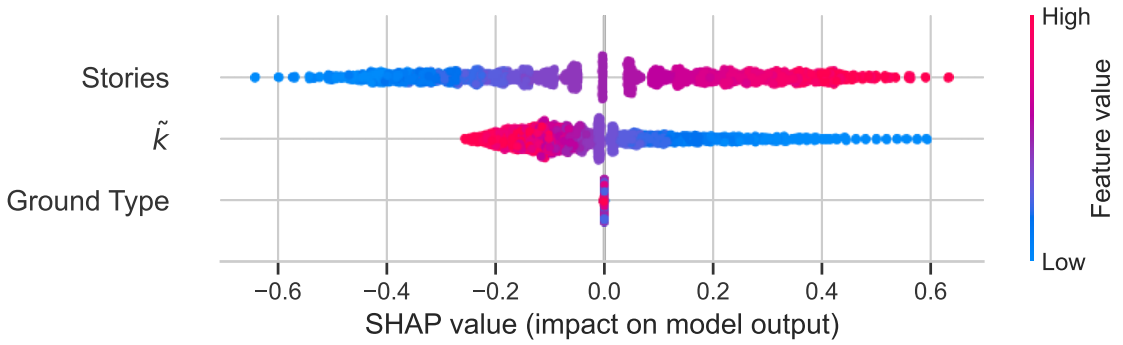


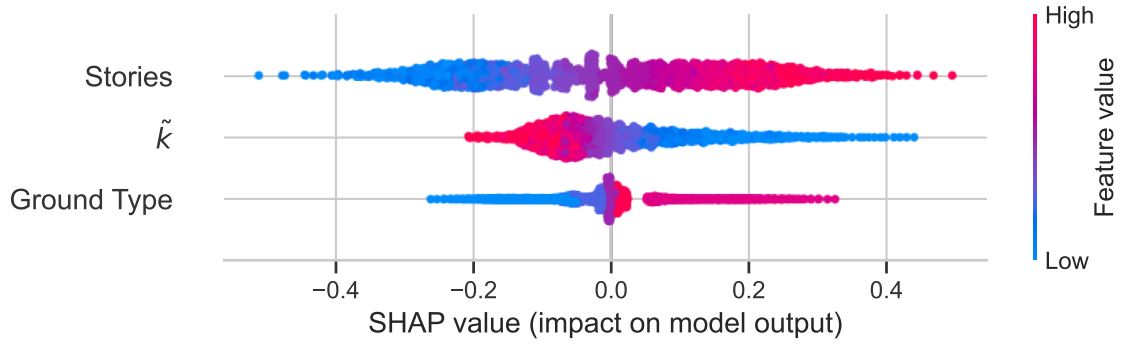
Figure 11. SHAP feature importance plot for the best-performing ML model.

6.2. Summary Plots

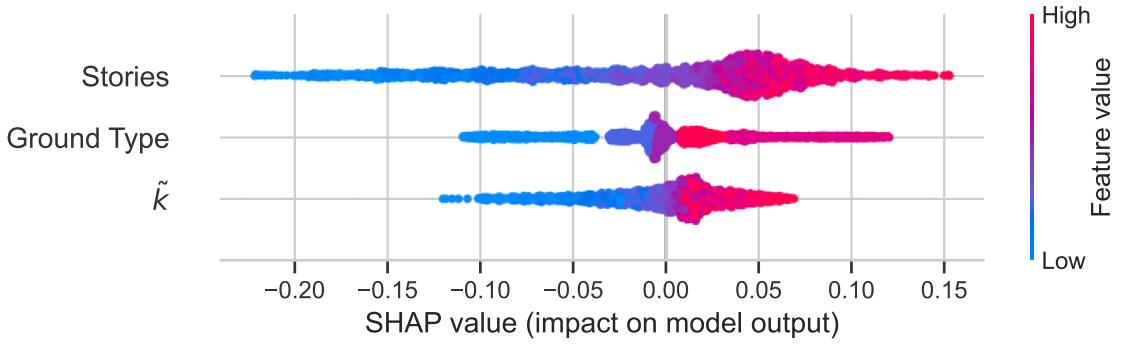
Although the feature importance plot is useful, it contains no information beyond the importance. For a deeper explanation of a machine learning model, additional informative plots would be needed. One of them is the so-called *summary* or *beeswarm* plot. A beeswarm plot visualizes all of the SHAP values in which the feature order (top to bottom) follows their importance to the prediction. On the vertical axis, the values are grouped by feature and the color of the points indicates the feature value ranging from low (blue) to high (red), for each group. Points with the same Shapley values for each feature are scattered vertically which subsequently forms their distribution. In Figure 12, the SHAP beeswarm plots of the best performing for each ML model are shown.



(**a**) T_1 model



(**b**) U_{top} model



(**c**) $\tilde{V}_b$ model

Figure 12. Summary plots showing the impact of all features on (**a**) T_1, (**b**) U_{top}, and (**c**) $\tilde{V}_b$ models.

It can be seen that for the number of stories, as the feature value increases, the SHAP values increase, too. This tells us that higher number of stories will lead to a higher predicted value for all models. In the case of the $\tilde{k}$ feature, we notice that as the feature value increases the SHAP values increase for the T_1 and U_{top} models, while in contrast for the same feature the SHAP values decrease in the case of the V_b model. As expected, the Ground Type feature has no impact on the predictions of the T_1 model, while it has an impact on the predicted U_{top} and V_b values.

7. Test Case Scenarios

We consider three test case scenarios for testing the effectiveness of the developed models and, in particular, the selected CatBoost prediction model. The first is a 3-story building, followed by a 8-story building and a 15-story building. The feature values for each scenario are presented in Table 5. The normalized stiffness ($\tilde{k}$) for each scenario is 2098.21, 5135.14, and 7169.81 s^{-2}, respectively. For practical reasons, we prefer to take k and m as independent parameters in the beginning and then calculate $\tilde{k}$, instead of working with $\tilde{k}$ from start, but it is essentially the same.

Table 5. Feature values for each test case scenario.

Scenario	Stories [-]	Mass (m) [kg]	Stiffness (k) [N/m]	Ground Type [-]	a_g [m/s^2]
1	3	112×10^3	235×10^6	B	0.32
2	8	185×10^3	950×10^6	A	0.24
3	15	265×10^3	1900×10^6	C	0.16

The results are presented in Table 6 for the three outputs, i.e., the fundamental period T_1, the displacement of the top story U_{top} and the base shear force V_b, for each scenario. In all cases, the prediction model managed to give results of very high precision with error values less than 3%. The maximum error value is only 2.93% corresponding to the shear force for the first scenario. It has to be noted that the model gives $\tilde{V}_b$, the normalized base shear force over the mass of each story of the building. By multiplying this with the mass m, we obtain the last column of the table which corresponds to the final base shear force V_b.

Table 6. Target values (actual and predicted) for each test case scenario. The absolute error is also provided.

		T_1 [s]	U_{top} [m]	V_b [N]
Scenario 1	Actual	0.308	0.0275	2.899×10^6
	Predicted	0.314	0.0285	2.816×10^6
	Absolute Error	1.95%	3.49%	2.85%
Scenario 2	Actual	0.475	0.0358	6.333×10^6
	Predicted	0.482	0.0365	6.336×10^6
	Absolute Error	1.47%	2.01%	0.05%
Scenario 3	Actual	0.733	0.0638	1.241×10^6
	Predicted	0.740	0.0650	1.241×10^6
	Absolute Error	0.95%	1.75%	0.53%

8. Web Application

The best performing ML models based on CatBoost, were used to develop an interactive web application. The GUI of the application is shown in Figure 13 for the input and predicted values of the first case scenario. It serves for rapid predictions of the dynamic response of multi-story buildings. More specifically, it can provide predictions of the fundamental eigenperiod, as well as of the roof top horizontal displacement and the shear base for the requested configurations of stories, mass, stiffness, and ground types. The web application is developed in Flask web framework and can be deployed in every platform with a Python environment with the required packages. The source code of the application is available at https://github.com/geoem/drsb-ml (accessed on 6 May 2023).

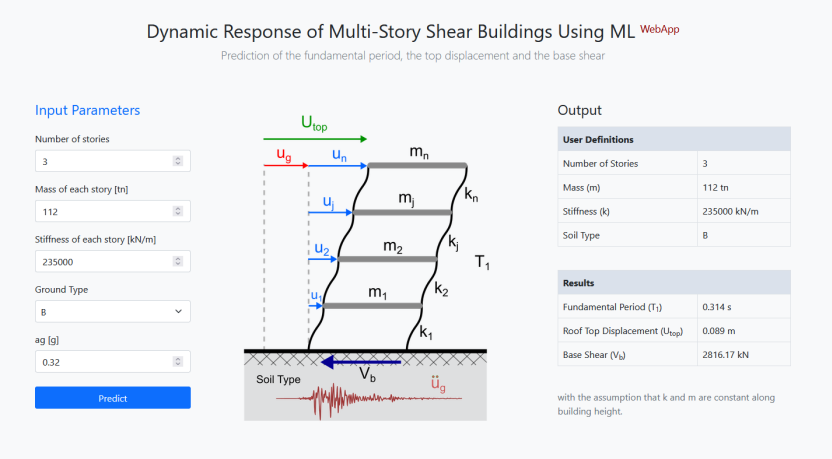


Figure 13. Web application GUI for rapid predictions of the dynamic response of multi-story shear buildings.

9. Conclusions

This paper presented the assessment of several ML algorithms for predicting the dynamic response of multi-story shear buildings. A large dataset of dynamic response analyses results was generated through standard sampling methods and conventional response spectrum modal analysis procedures of multi-DOF structural systems. Then, an extensive hyperparameter search was performed to assess the performance of each algorithm and identify the best among them. Of the algorithms examined, CatBoost came first in accuracy with acceptable fit and predicted times for most of the cases, while Ridge Regressor took the trophy for being the fastest to fit the data. Overall, CatBoost appeared to be the best performing algorithm, although for the case of the normalized shear base model, the Random Forest algorithm exhibited slightly better performance.

The results of this study show that ML algorithms, and in particular CatBoost, can successfully predict the dynamic response of multi-story shear buildings, outperforming traditional simplified methods used in engineering practice in terms of speed, with minimal prediction errors. The work demonstrates the potential of ML techniques to improve seismic performance assessment in civil and structural engineering applications, leading to more efficient and safer designs of buildings and other structures. Overall, the use of ML algorithms in the dynamic analysis of structures is a promising approach to accurately predict the dynamic behavior of complex systems.

The study has also some limitations that need to be highlighted and discussed. First of all, the analysis is only elastic and the behavioral factor q of the design spectrum of EC8 takes the fixed value of 1 throughout the study. In addition, damping has been considered with a fixed value of 5%, while the stiffness and mass of each story remains constant along the height of the building. The extension of the work in order to account for these limitations is a topic of interest which will be investigated in the future by adding extra features to the ML model.

Author Contributions: Conceptualization, M.G. and V.P.; methodology, M.G. and V.P.; software, M.G.; validation, M.G. and V.P.; formal analysis, M.G.; investigation, M.G.; resources, M.G.; data curation, M.G.; writing—original draft preparation, M.G.; writing—review and editing, M.G. and V.P.; visualization, M.G.; supervision, V.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The source code of the web application is available at https://github.com/geoem/drsb-ml (accessed on 6 May 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Solorzano, G.; Plevris, V. Computational intelligence methods in simulation and modeling of structures: A state-of-the-art review using bibliometric maps. *Front. Built Environ.* **2022**, *8*, 1049616. [CrossRef]
2. Georgioudakis, M.; Plevris, V. A Combined Modal Correlation Criterion for Structural Damage Identification with Noisy Modal Data. *Adv. Civ. Eng.* **2018**, 3183067.
3. Lagaros, N.D.; Plevris, V.; Kallioras, N.A. The Mosaic of Metaheuristic Algorithms in Structural Optimization. *Arch. Comput. Methods Eng.* **2022**, *29*, 5457–5492.
4. Plevris, V.; Lagaros, N.D.; Charmpis, D.; Papadrakakis, M. Metamodel assisted techniques for structural optimization. In Proceedings of the First South-East European Conference on Computational Mechanics (SEECCM-06), Kragujevac, Serbia, 28–30 June 2006 ; pp. 271–278.
5. Papadrakakis, M.; Lagaros, N.D.; Tsompanakis, Y.; Plevris, V. Large scale structural optimization: Computational methods and optimization algorithms. *Arch. Comput. Methods Eng.* **2001**, *8*, 239–301. [CrossRef]
6. Lagaros, N.; Tsompanakis, Y.; Fragiadakis, M.; Plevris, V.; Papadrakakis, M. Metamodel-based Computational Techniques for Solving Structural Optimization Problems Considering Uncertainties. In *Structural Design Optimization Considering Uncertainties*; Tsompanakis, Y., Lagaros, N., Papadrakakis, M., Eds.; Taylor & Francis: Abingdon, UK, 2008; Chapter 21, pp. 567–597.
7. Lu, X.; Plevris, V.; Tsiatas, G.; De Domenico, D. Editorial: Artificial Intelligence-Powered Methodologies and Applications in Earthquake and Structural Engineering. *Front. Built Environ.* **2022**, *8*, 876077. [CrossRef]
8. Xie, Y.; Sichani, M.E.; Padgett, J.E.; DesRoches, R. The promise of implementing machine learning in earthquake engineering: A state-of-the-art review. *Earthq. Spectra* **2020**, *36*, 1769–1801. [CrossRef]
9. Zhang, Y.; Burton, H.V.; Sun, H.; Shokrabadi, M. A machine learning framework for assessing post-earthquake structural safety. *Struct. Saf.* **2018**, *72*, 1–16. [CrossRef]
10. Nguyen, H.D.; Dao, N.D.; Shin, M. Prediction of seismic drift responses of planar steel moment frames using artificial neural network and extreme gradient boosting. *Eng. Struct.* **2021**, *242*, 112518. [CrossRef]
11. Sadeghi Eshkevari, S.; Takáč, M.; Pakzad, S.N.; Jahani, M. DynNet: Physics-based neural architecture design for nonlinear structural response modeling and prediction. *Eng. Struct.* **2021**, *229*, 111582. [CrossRef]
12. Abd-Elhamed, A.; Shaban, Y.; Mahmoud, S. Predicting Dynamic Response of Structures under Earthquake Loads Using Logical Analysis of Data. *Buildings* **2018**, *8*, 61. [CrossRef]
13. Gharehbaghi, S.; Gandomi, M.; Plevris, V.; Gandomi, A.H. Prediction of seismic damage spectra using computational intelligence methods. *Comput. Struct.* **2021**, *253*, 106584. [CrossRef]
14. Kazemi, F.; Asgarkhani, N.; Jankowski, R. Machine learning-based seismic response and performance assessment of reinforced concrete buildings. *Arch. Civ. Mech. Eng.* **2018**, *23*, 94. [CrossRef]
15. Kazemi, F.; Jankowski, R. Machine learning-based prediction of seismic limit-state capacity of steel moment-resisting frames considering soil-structure interaction. *Comput. Struct.* **2023**, *274*, 106886. [CrossRef]
16. Wakjira, T.G.; Rahmzadeh, A.; Alam, M.S.; Tremblay, R. Explainable machine learning based efficient prediction tool for lateral cyclic response of post-tensioned base rocking steel bridge piers. *Structures* **2022**, *44*, 947–964. [CrossRef]
17. Montuori, R.; Nastri, E.; Piluso, V.; Todisco, P. A simplified performance based approach for the evaluation of seismic performances of steel frames. *Eng. Struct.* **2020**, *224*, 111222. [CrossRef]
18. *EN 1998-1 (Eurocode 8)*; Design of Structures for Earthquake Resistance—Part 1: General Rules, Seismic Actions and Rules for Buildings. European Committee for Standardization: Brussels, Belgium, 2004.
19. Hoerl, A.E.; Kennard, R.W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics* **1970**, *12*, 55–67. [CrossRef]
20. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
21. Friedman, J.H. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
22. Prokhorenkova, L.O.; Gusev, G.; Vorobev, A.; Dorogush, A.V.; Gulin, A. CatBoost: Unbiased boosting with categorical features. In Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montreal, QC, Canada, 4–6 December 2018; Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; pp. 6639–6649.
23. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
24. Plevris, V.; Solorzano, G.; Bakas, N.; Ben Seghier, M.E.A. Investigation of performance metrics in regression analysis and machine learning-based prediction models. In Proceedings of the 8th European Congress on Computational Methods in Applied Sciences and Engineering, Oslo, Norway, 5–9 June 2022. [CrossRef]
25. Lundberg, S.M.; Lee, S.I. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.