Challenge Journal of Structural Mechanics

Research Article

# Ultimate drift ratio prediction of steel plate shear wall systems: a machine learning approach

Muhammed Gürbüz [a,*], İlker Kazaz [a]

[a] *Department of Civil Engineering, Erzurum Technical University, 25050 Erzurum, Türkiye*

**ABSTRACT**

Predicting the ultimate drift ratio of steel plate shear wall (SPSW) systems is important for ensuring the structural integrity and performance of these systems under lateral loads. In this study, machine learning models are developed for predicting the ultimate drift ratio based on various design parameters using data from previous research on SPSW systems. These design parameters include the panel aspect ratio, column flexibility parameter, axial load ratio, web plate thickness and stiffness of horizontal and vertical boundary elements. A range of machine learning models is considered, including Random Forest, Lasso, Gradient Boosting, XGBoost, Adaboost, Artificial Neural Network and a stacked regressor. The models are trained and evaluated using data from 292 different SPSW models, and their performance is compared based on the R-squared value, root mean squared error (RMSE), and evaluation time. The results of this study demonstrate the effectiveness of machine learning techniques for predicting the ultimate drift ratio of SPSW systems. The results of this study show that machine learning techniques effectively predict the ultimate drift ratio of SPSW systems. Among the models considered, the ANN model achieved the highest $R^2$ value, with a value of 0.94.

**ARTICLE INFO**

*Article history:*
Received 13 September 2023
Revised 2 November 2023
Accepted 25 November 2023

*Keywords:*
Steel plate shear walls
Finite element analysis
Machine learning
Artificial neural network
Stacked regressor



## 1. Introduction

Structural engineering involves the analysis and design of load-bearing structures that can be particularly challenging in complex structural systems under extreme actions exhibiting highly nonlinear behavior. Traditional structural analysis and design methods involve the study and application of mathematical and physical principles to evaluate the performance and behavior of load-bearing structures. These methods are used to predict the response of structures to various types of loads, such as gravity, wind, and earthquakes, and to design safe and efficient structures. However, these methods may require a time-consuming calibration process to accurately predict the behavior of systems under extreme loading conditions, particularly for those that exhibit highly nonlinear behavior. Additionally, the complexity and uncertainty in these methods can make them challenging to implement in practice, particularly for complex structural systems. However, machine learning (ML) allows for the creation of computer models that can learn and identify patterns from a set of data, potentially offering more efficient and effective solutions for structural engineering problems.

Artificial intelligence (AI) is a computer science field aiming to replicate human cognition and reasoning capabilities through symbol manipulation and structured knowledge bases. Machine learning (ML) is a subfield of AI that focuses on teaching computers to make predictions from data and algorithms without being explicitly programmed. ML algorithms can be classified into unsupervised and supervised learning. Unsupervised learning is a type of ML in which an algorithm is trained on an unlabeled dataset. It is used to discover patterns and relationships in the data without prior knowledge of the output. Clustering, a subcategory of unsupervised learning, involves grouping data points into clusters based on similarity. Dimensionality reduction, another subcate-

* *Corresponding author*. E-mail address: muhammed.gurbuz@erzurum.edu.tr (M. Gürbüz)
ISSN: 2149-8024 / DOI: https://doi.org/10.20528/cjsmec.2024.02.001

gory of unsupervised learning, involves reducing the number of variables in a dataset while preserving the important information. These techniques can be useful in structural engineering tasks such as identifying similar structures or simplifying complex models. Supervised learning is a type of ML in which an algorithm is trained on a labelled dataset. It is commonly used for regression and classification problems in which the output is continuous or discrete. Regression analysis, a subcategory of supervised learning, involves predicting a continuous output based on one or more input variables. It is widely used in various fields, including structural engineering, to make predictions about structural behavior, such as strength and stiffness, or to identify relationships between variables.

In structural engineering, various ML algorithms have been applied to multiple tasks, including predicting building responses to seismic loads, analyzing structural behavior under extreme actions, and optimizing structural design parameters. These algorithms include neural networks (NN), which use artificial neural connections to model relationships between input and output data; decision trees (DT), which use a tree-like model of decisions to identify the most appropriate course of action; regression analysis (RA), which is used to predict continuous values based on a given set of input variables; support vector machines (SVM), which classify data by finding the hyperplane that maximally separates different classes; random forests (RF), which are ensembles of decision trees that use random sampling to improve the prediction accuracy of the model; and boosting algorithms (BA), which combine multiple weak models to create a more accurate and robust model.

Many pioneering works in structural engineering, including structural analysis and design, have effectively employed the machine learning technique. Adeli and Yeh (1989) completed one of the first examples of structural design work on beam design. Hajela and Berke (1991) used neural networks to study force-displacement relationships in static structural analysis in the following years. They applied two different network algorithms to the problem and examined the effects of parameters such as learning rate on the results. Xu et al. (1992) created neural network-based methods for autonomously detecting structural damage caused by a variety of variables such as routine operations, accidents, deterioration, or natural events. They concluded that the use of the neural network is promising for structural damage assessment.

Large experimental or numerical datasets are often required to confirm machine learning's performance in estimating the behavior of structural systems. One of the reasons why machine learning is becoming more popular in structural engineering is because the needed datasets are considerably larger than in the past. The survey by Thai (2022) provides a comprehensive review of the growing applications of machine learning algorithms. The research investigates how machine learning can be applied to ascertain the mechanical characteristics and mix design of concrete. It delves into examining both individual elements and entire structures constructed from steel, concrete, and composite materials. Sun et al. (2021) review the application of machine learning in structural design and performance assessment. The paper also discusses the challenges and future opportunities for integrating machine learning into structural engineering practice. Xie et al. (2020) studied the progress and challenges of implementing machine learning in earthquake engineering. The review encompasses the utilization of machine learning across four primary domains system identification, seismic analysis, earthquake engineering and damage detection, seismic fragility assessment, and structural control for earthquake mitigation. The study also discusses the challenges and future research needs for machine learning in earthquake engineering, including embracing new data sharing and sensor technologies, implementing advanced machine learning techniques, and developing physics-guided machine learning models.

There are also studies to predict structural drift limits using machine learning methods. Inel (2007) used experimental data from 237 rectangular columns to create an artificial neural network (ANN) model and compared the ultimate displacement estimates from the model to existing semi-empirical and empirical models. The results showed that the ANN model performed well and proved the feasibility of using ANN models for estimating the deformation of RC columns. Darain et al. (2015) investigated the deflection behavior of strengthened RC beams using adaptive neuro-fuzzy prediction. According to the study, the projected deflection and crack width nearly match the outcomes of the experiments, confirming the accuracy of the model. Another study by Kalman et al. (2013) evaluated the earthquake performance of RC frames with masonry wall infill using NN (neural network). The study results showed that neural networks trained on the database could be utilized to estimate the seismic behavior of framed-masonry structural elements. Luo and Paal (2019) proposed a local machine-learning model for predicting the drift capacity of reinforced concrete (RC) columns. The model was trained and tested using a database comprising 160 circular reinforced concrete columns. It was then contrasted with well-known existing global and local learning methods as well as a conventional empirical equation. The outcomes indicated that the suggested model is a prospective method for improving the predicting of drift capacity. Nguyen et al. (2021) studied the seismic drift behavior of planar steel moment frames using ANN and Extreme Gradient Boosting (XGboost). The authors stated that the XGboost algorithm better predicted the responses of steel frames than the ANN model. The authors also designed a user interface to predict steel frame drift response using XGboost.

In this study, the performance of different machine learning-based regression models in predicting the drift capacity of steel plate shear walls (SPSW) is evaluated using data from a previous study (Gürbüz and Kazaz 2022a). SPSW systems are widely used in the construction of high-rise buildings and other structures subjected to lateral loads such as earthquakes and wind. Steel web shear walls have a more sophisticated system mechanism than conventional steel frames. A steel web plate in

such systems transfers additional loads on the horizontal and vertical boundary members (beams and columns). The lower and higher-level steel panels transmit opposing loads to the beam members between adjacent floors. As a result, the top anchor beam experiences an unbalanced load. Moreover, the steel panel imposes an inward pull on the columns, possibly leading to unexpected column buckling. Steel panel-induced loads have resulted in various steel frame collapse mechanisms. These collapse processes under monotonic and cyclic loading cases were investigated by Gürbüz and Kazaz (2022a, 2022b). The performance of these systems under lateral loads depends on various design parameters, including the panel aspect ratio, column flexibility parameter, axial load ratio, web plate thickness, and more. Previously, 292 different SPSW models were designed and analyzed by Gürbüz and Kazaz (2022a) using the finite element method to explore the interactive effect of various design parameters on the drift capacity and shear force distribution of SPSW systems. Accurate prediction of the ultimate drift ratio of SPSW systems is essential for ensuring the safety and reliability of these structures. Various past studies have investigated the interactive effect of design parameters on the drift capacity and shear force distribution of SPSW systems using analytical and experimental methods. However, these studies often examined only a few design parameters at a time, making it difficult to fully understand the complex interaction of all the parameters on the behavior of SPSW systems. The design parameters included panel aspect ratio, column flexibility factor (a measure of the stiffness of the columns), axial load ratio, web plate thickness, stiffness of horizontal and vertical boundary elements. The ultimate drift ratio of each model was used as the response variable in the machine-learning models. The ultimate drift ratio is a measure of the maximum displacement of the structure under lateral load and is an essential consideration in the design of SPSW systems as it affects the overall stability and performance of the structure.

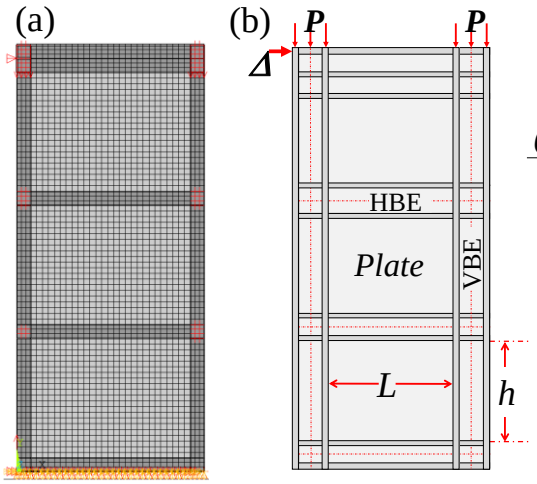
This study aims to fill the gap in the literature by developing machine learning model algorithms to predict the ultimate drift ratio of SPSW systems based on the design parameters. Many machine learning models, including Random Forest, Lasso, Gradient Boosting, XGBoost, Adaboost, and a stacked regressor, are considered. In addition, the performance of an artificial neural network (ANN) model is also investigated. ANNs are a type of machine learning algorithm that mimics the structure and function of the human brain and can be used for both regression and classification tasks. By comparing the performance of these different models, the researchers can determine the most effective approach for predicting the behavior of these types of structures.

The details of the algorithms used in this study are given in the following sections. The performance of these models is compared in terms of the $R^2$ value, root-mean-squared error (RMSE), and evaluation time. The results of this study provide a comprehensive understanding of the behavior of SPSW systems and may be useful for predicting the ultimate drift ratio of these systems in the design process.

## 2. Data

Gürbüz and Kazaz (2022a) used a verified finite element modeling technique to investigate the behavior of the 3-story and moment-framed steel plate shear walls. 292 models were created by changing design parameters within specific ranges and analyzed under monotonic loading using the finite element analysis software ANSYS (2016).

To create variations of SPSW models, plate thickness, panel aspect ratio, stiffness of boundary elements, and axial load ratio on vertical boundary element (VBE) were considered. The models contained both geometric and material non-linearities. Fig. 1 shows a typical three-story SPSW model that illustrates and summarizes the study's main parameters. In the model, to increase the resistance of the top beam against the bending action of internal steel plate forces, an additional beam was attached to the top beam, and the beams were assumed to be constrained (share the same nodes) along the adjoining beam flanges as shown in Fig. 1(a). The horizontal load was applied at the top left nodes of the models as shown in Fig. 1(b).




| $h$ (mm) | $L$ (mm) | $t_w$ (mm) | $I_c / I_b$ | $P / P_{cr}$ |
|---|---|---|---|---|
| 3000 | 2000 | 2 | 1 | 0.1 |
| | 3000 | 3 | 2 | 0.2 |
| | 4000 | 4 | 3 | 0.3 |
| | 5000 | 5 | 4 | |
| | 6000 | 6 | 5 | |

**Fig. 1.** (a) A representative finite element model of SPSWs used in the study; (b) Typical steel plate shear wall configuration.

All story-level intersections of beams and columns were defined as the locations for lateral support. At all story levels, out-of-plane displacements of the nodes in intersection regions were constrained.

The chosen panel lengths are 2000, 3000, 4000, 5000, and 6000 mm, and the steel panel height $h_s$ is constant at 3000 mm in all variants. In this manner, plate aspect ratios - the ratio of panel length to height $L/h$ - between 0.67 and 2 were considered. As previously observed, narrow steel plate shear walls functioned adequately and displayed ductile hysteric behavior, which can be compared to those with larger aspect ratios (Li and Tsai 2008). Thus, models with a low panel aspect ratio of 0.67 were also examined. The chosen plate thicknesses $t_w$ are 3, 4, 5, and 6.

The SPSW behavior was investigated using parameters plate thickness, column flexibility parameter ($\omega_i$), the moment of inertia ratio of beam and column ($I_c/I_b$), panel aspect ratio($L/h$), axial load ratio on columns($P/P_{cr}$), beam flange width–beam height ratio ($b/h_b$) the beam flange width to beam height ratio ($b_f/h_b$), the beam flange width-to-thickness ratio ($b_f/2t_f$), beam web area and column web area.

Seven different types of failure modes were identified for SPSWs. These failure modes were studied in terms of variables, including the column flexibility factor, plate thickness, and $L/h$ in general. Fig. 2 displays the identified SPSW failure modes and a representative force-drift curve for each group. Each group was carefully analyzed in the relevant research, and an investigation on the parameters that lead to failure and the failure sequence was conducted.

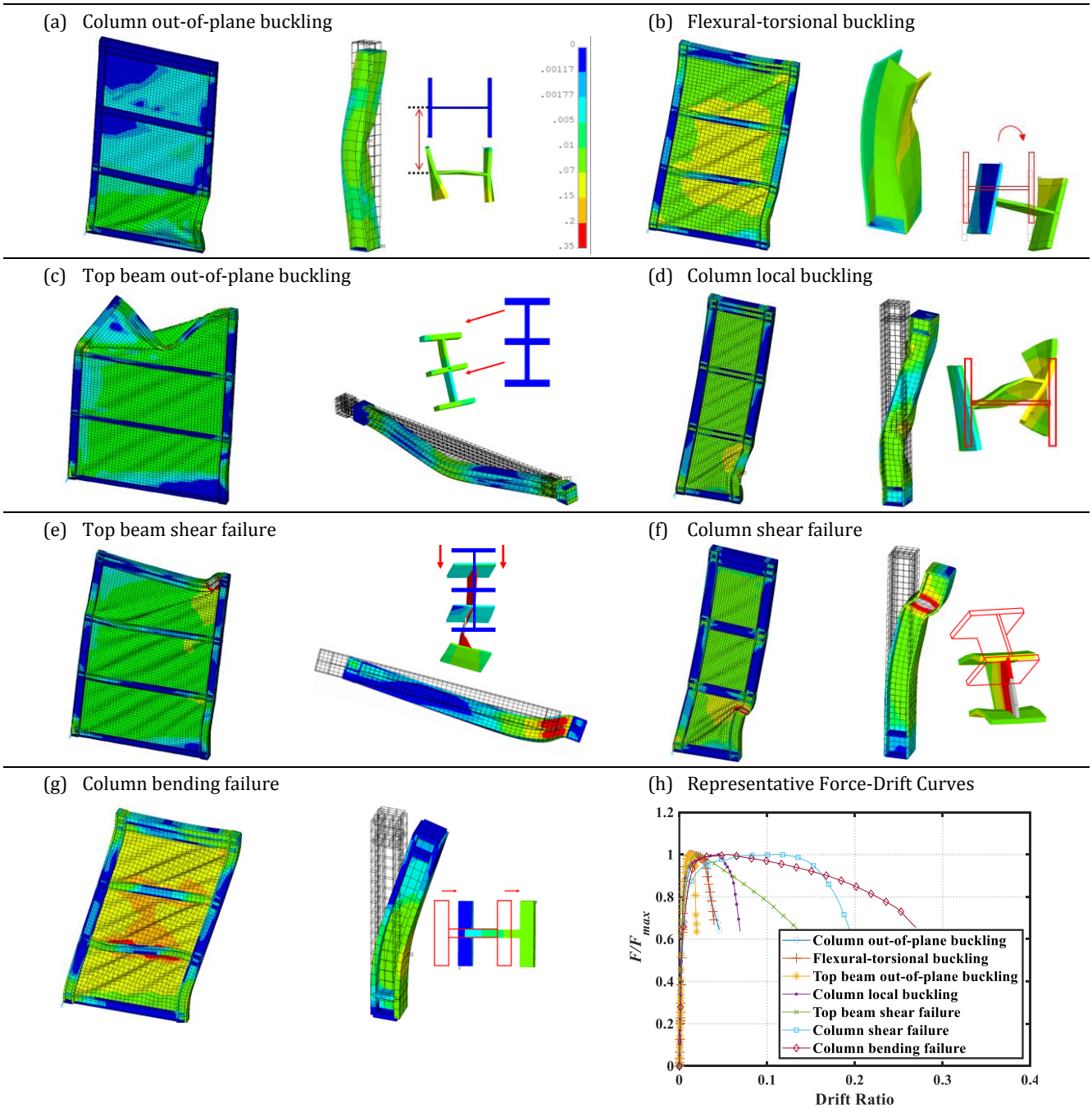



**Fig. 2.** Different failure modes of the SPSW models and representative Force-Drift ratio curves of each failure group.

The definition of the ultimate drift ratio ($DR_u$) of SPSWs varies for different models. When the force-drift curves are assessed, some groups show no noticeable drop, while others have a system that clearly fails as displayed in Fig. 3. Therefore, it could be claimed that this difference in how the ultimate drift ratio is defined is necessary.

The point where the base story shear force drops to 85% of its maximum value on the load-displacement curve is the ultimate drift ratio for the group that failure is observable (Fig. 3(a)). In a different group that is represented by the force-drift curve in Fig. 3(b), it can be seen that the system can still sustain the load at 85% of its maximal strength, but after a particular drift, it loses its ability to resist force. The ultimate drift ratio for these models is referred to as the location of the break on the force-displacement curve. The ultimate drift limit, however, is taken as the drift level corresponding to 50% of the peak shear force, such as in high axial load instances where the force-displacement curve steadily decreases after peak force (Fig. 3(c)).

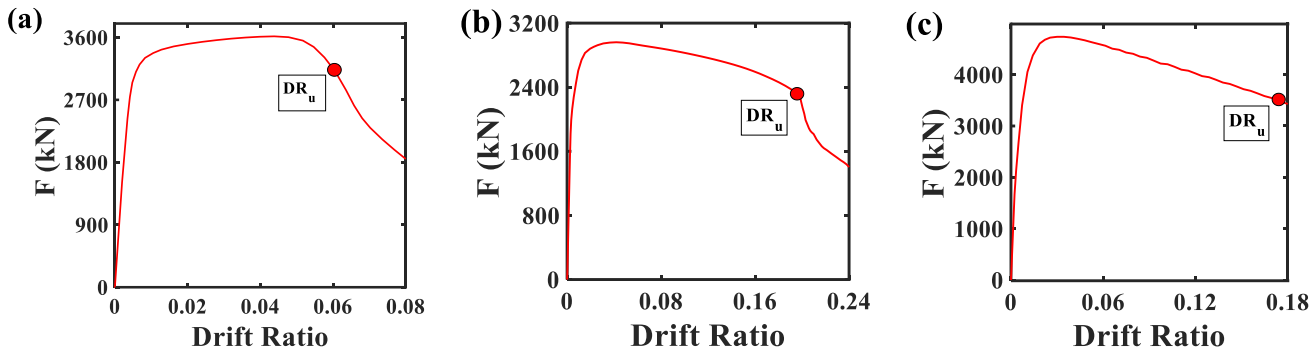



**Fig. 3.** Different definitions of the ultimate drift ratio in Gürbüz and Kazaz (2022a).

In ensuring data quality and preparing the dataset for model training, several crucial data preprocessing steps were employed. This included an extensive data cleaning process to handle outliers and anomalies, normalization techniques to standardize feature scales, and imputation methods to address missing values.

## 3. Method

The dataset employed in this study is composed of numerical data that simulates the behavior of steel plate shear walls with moment frames under extreme loading conditions. This dataset was created specifically to explore the performance of the SPSW structures under these high-loading conditions. By providing the data to feed the machine learning models, this study also investigates the parameter effects on the overall behavior of the structures.

The study employed a Stacking Regressor, which combines multiple base models to improve the prediction accuracy of the drift behavior of Steel Plate Shear Walls (SPSWs) under monotonic loading. A variety of base models were defined, including Gradient Boosting, Lasso, Random Forest, XGBoost, ANN and AdaBoost was selected due to their capability to deal with nonlinear and complex data and used as input to the Stacking Regressor. This ensemble approach aims to leverage the strengths of each individual model to achieve improved performance.

The choice of model hyperparameters is a critical issue in machine learning models. When creating models, different values are given to parameters like the learning rate and the number of estimators, which directly impact the prediction accuracy of the models. These parameters can be chosen randomly or optimized using different methods. One of these methods is the grid search technique finding the ideal set of hyperparameters for a particular model. To achieve this, a model is trained using various combinations of hyperparameters, and its effectiveness is assessed using test data. The ideal setting is then determined by the combination of hyperparameters that produces the best performance. Many supervised machine learning tasks use grid search. The grid search can be performed using sklearn's GridSearchCV class. This section provides information about each machine learning algorithm used in this study.

### *3.1. AdaBoost*

The AdaBoost algorithm was employed as one of the base models in the stacking regressor. AdaBoost, which stands for Adaptive Boosting, is a type of ensemble method that combines a number of weak learners to form a strong learner. Freund and Schapire (1997) introduced it in 1997 and since then has been widely used in various applications. The basic idea behind AdaBoost is to iteratively improve the performance of a weak model by giving more weight to the misclassified examples in the training data.

In this study, the AdaBoost algorithm is implemented using the Sklearn library in Python. Scikit-learn (Sklearn) is an open-source machine learning library for the Python programming language. It provides a wide range of tools and techniques for machine learning and statistical modeling, including classification, regression, clustering, and dimensionality reduction ("scikit-learn: machine learning in Python - scikit-learn 1.2.0 documentation" n.d.). The AdaBoostRegressor class is used to create the model, which takes several parameters such as the number of estimators, the learning rate, and the random state. The number of estimators refers to the num-

ber of weak learners to be used, the learning rate determines the weight of each estimator in the final prediction, and the random state is used to initialize the random number generator for reproducibility.

The number of estimators is set to 200, 400, and 600, the learning rate is set to 0.001, 0.005, and 0.01, and the random state is set to 0 for the grid search. Finally, the learning rate is set to 0.01 and the number of estimators is 600.

### *3.2. Gradient Boosting*

Gradient Boosting is an ensemble learning technique that is an iterative method to optimize a differentiable loss function using the gradient descent algorithm. The idea behind gradient boosting is to train a set of weak models (such as decision trees) and combine them to form a stronger model. Each tree grown during the iteration fits the negative gradient of the loss function (or residual) concerning the current ensemble predictions. After each iteration, the negative gradient is computed as the difference between the true target values and the current ensemble predictions. A typical loss function used in gradient boosting is mean squared error (MSE) for regression problems is given as,

$$MSE = \frac{1}{n}\sum(y_i - \hat{y}_i)^2 \quad (1)$$

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

In this study, the Gradient Boosting algorithm is implemented using the sklearn library in Python. The GradientBoostingRegressor class is used to create the model, which takes several parameters such as learning rate, loss function, maximum depth of the trees, the minimum number of samples required to split a node, and the number of estimators. Using the GridSearchCV class, the learning rate is set to 0.01. The loss function is set to 'ls' (least squares), which minimizes the mean squared error between the predicted and actual response values. The maximum depth of the trees is 6, the minimum samples required to split a node is 4, and the number of estimators is 600.

### *3.3. Random Forest*

Random forest is a popular algorithm for many machine-learning tasks, including classification and regression. The idea behind it is to construct multiple decision trees and combine their predictions with improving the overall performance of the model. Each tree in a random forest is built using a different subset of the training data and a different subset of the features. The final prediction is made by averaging or voting the predictions of all the trees in the forest. Fig. 4 depicts a schematic of the Random Forest working algorithm.

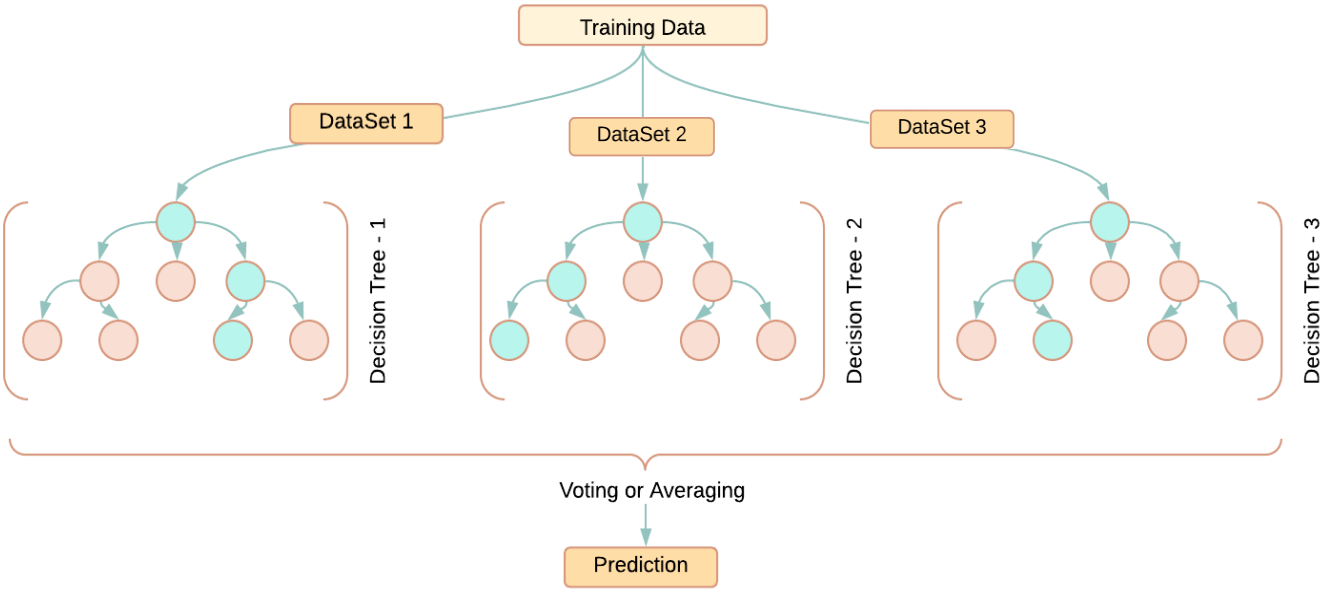



**Fig. 4.** Random Forest algorithm.

The random forest algorithm was first introduced by Breiman (2001). The idea of building a model using multiple decision trees was motivated by the problem of high variance in individual decision trees, which often resulted in poor generalization performance. By averaging the predictions of multiple trees, random forest is able to reduce the variance and improve the overall performance of the model.

In terms of implementation, one can use the RandomForestClassifier and RandomForestRegressor classes from the scikit-learn library to train and evaluate a random forest model. These classes provide a simple and easy-to-use interface for training and evaluating random forest models and provide a variety of parameters for fine-tuning the model performance. Grid search and cross-validation techniques can also be used to tune the parameters of the random forest model.

The RandomForestRegressor class is used to create the model, which takes several parameters such as the maximum depth of the trees, the minimum number of

samples required to split a node, the number of estimators and the random state. Using the GridSearchCV class, the maximum depth of the trees is set to 8, the minimum number of samples required to split a node is set to 4, the number of estimators is set to 400 and the random state is set to 0. These hyperparameters are used to initialize the Random Forest model, which is then trained on the provided dataset to make predictions. It is important to note that these values are not necessarily optimal for every dataset and problem and that tuning these hyperparameters is crucial to achieve good performance.

### *3.4. XGBoost*

XGBoost, also known as eXtreme Gradient Boosting, is a powerful gradient-boosting algorithm that is specifically designed to be efficient and fast. It was first introduced by Chen and Guestrin (2016). It is an optimized implementation of the gradient boosting framework, which sequentially combines decision trees to improve the overall accuracy of the model. XGBoost is a popular algorithm in machine learning competitions due to its ability to handle large and high-dimensional data and to be robust in handling missing data.

The algorithm works by fitting an ensemble of decision trees, with each new tree trained to correct the errors made by the previous one. It uses several regularization and optimization techniques to improve the speed and performance of the algorithm. Chen and Guestrin (2016) stated that it is ten times faster than existing solutions. The output variable is given in the following equation:

$$\hat{y}_i = \sum_{k=1}^{k} f_k(x_i), f_k \in \mathrm{F} \tag{2}$$

where, $\hat{y}$ and $x_i$ are the output and input variables, $K$ is the number of trees, $f$ is the score function and $F$ is the possible regression or classification tree set. The regularized objective function is:

$$L(\emptyset) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{3}$$

where, $l$ is a differentiable convex loss function and the second term Ω penalizes the complexity of the model.

The XGBoost algorithm is implemented using the XGBRegressor class of the XGBoost library. It takes several parameters, such as the learning rate, maximum depth of the trees, and the number of estimators. Using the GridSearchCV class, the learning rate is set to 0.01, the maximum depth of the trees is 8, and the number of estimators is 600. These hyperparameters are used to initialize the XGBoost model, which is then trained on the provided dataset to make predictions.

### *3.5. Lasso Regression*

The Lasso algorithm, which stands for Least Absolute Shrinkage and Selection Operator, was first introduced by Robert Tibshirani (1996). Apart from the algorithms given above, Lasso is a linear model that uses L1 regularization, which is a technique that constrains the sum of the absolute values of the coefficients. The Lasso algorithm was developed as an extension of the least squares method, a commonly used technique for fitting a linear model to a dataset. However, unlike least square, which uses L2 regularization, Lasso uses L1 regularization, resulting in sparse solutions where some of the coefficients are zero. This makes Lasso particularly useful for feature selection, as it can automatically select a subset of the most important features for the model.

The Lasso algorithm has been widely used in various fields such as statistics, machine learning, and applied sciences. It is advantageous in high-dimensional settings where the number of features is larger than the number of samples. Since its introduction, many extensions and variations of the Lasso algorithm have been developed, such as the elastic net, which is a combination of L1 and L2 regularization, and the adaptive Lasso, which adapts the L1 penalty to the data.

In this study, the alpha value is set to 0.001, which controls the regularization strength. A larger value will make the model more conservative, while a smaller value will make the model more flexible and prone to overfitting.

### *3.6. Stacked Regressor*

The StackingRegressor class can combine the predictions of multiple base estimators and use them for training a higher-level meta-model. The base estimators are trained in parallel, and their predictions are combined using a blending process, which can be specified through the blender parameter. The meta-model is then trained on the blended predictions of the base estimators.

In this case, the StackingRegressor object is a model that combines the predictions of the Gradient Boosting model, Lasso model, Random Forest Model, and XGBoost model using the RidgeCV model as the meta-model. This can potentially improve the performance of the resulting model compared to using any of the base estimators alone.

The stacked regressor model, however, came after to the XGBoost model in this investigation. The stacked regressor model's $R^2$ value is 0.79, and the evaluation process took 11.62 sec in total. Among all the models, this evaluation period is the longest. This is a predictable outcome, given that using a new meta-model and combining additional regression models takes some time. However, the stacked regressor results being less accurate than the XGBoost model is generally not expected. It is worth considering whether the ensemble model is appropriate for the data. Ensemble models can be powerful, but they may not always be the best choice for a particular dataset. The chosen parameters might have high relations to each other, resulting in the ensemble model being unable to capture the full range of prediction patterns and performing worse than the individual models. A correlation investigation was conducted and the results are presented in the results section.

### *3.7. Artificial Neural Network*

Artificial neural networks (ANNs) are a class of machine learning models that are inspired by the structure and function of biological neural networks. The ANN

consists of numerous layers of neurons, which process and transmit information. ANNs are capable of learning from data and can be used for a variety of tasks. The working mechanism of an ANN involves the processing of input data through the various neuron layers, each of which applies a set of weights and biases to the input and produces an output. An example of the ANN algorithm is given in Fig. 5(a). The output of one layer serves as the input to the next layer. The weights and biases are adjusted during the training process to optimize the performance of the network

The ANN model is created in this study using the Neural Network Toolbox in MATLAB ("Deep Learning Toolbox - MATLAB" n.d.). The given code creates a loop that runs 60 times. In each loop iteration, a neural network with a different number of neurons in the hidden layer is created and trained using the fitnet and train functions. The training ratio is set to 70%. The validation ratio is 20% while the test ratio is 10%. It then sets the proportion of data for training, validation and testing and trains the network using the train data. Next, the code calculates the Root Mean Squared Error (RMSE) for the training and validation sets for each hidden layer size. After all, the hidden layer sizes have been trained and the errors have been calculated, the code used in this study finds the index of the minimum RMSE value on the validation set and prints the optimal number of neurons in the hidden layer. The code then retrains the network with an optimal number of neurons with a training ratio of 70%, a validation ratio of 20% and a testing ratio of 10%. Then the code calculates the RMSE and relative RMSE of the training and validation dataset. An image created by the MATLAB Neural Network tool that shows the number of inputs, the number of optimal hidden layers and the output is given in Fig. 5(b).

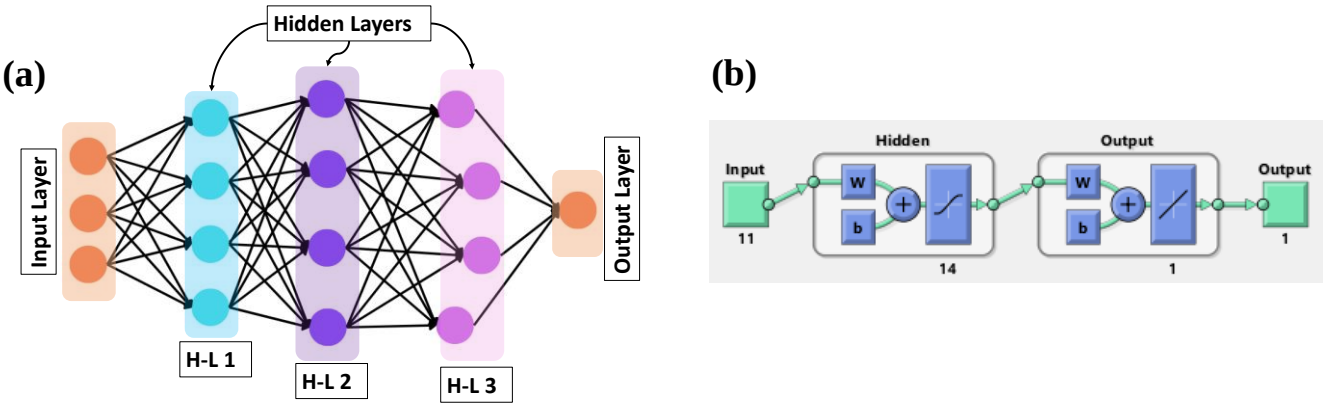



**Fig. 5.** (a) ANN architecture; (b) An architecture created by MATLAB Neural Network tool.

In each run, the optimal number of neurons takes a different value. The reason is that the hidden layer changes in each run since the training and validation sets are selected randomly at the beginning of each run. This means that the data on which the ANN is trained and evaluated will be different in each run, and therefore the optimal number of neurons in the hidden layer that leads to the best performance on that specific data will also be different. The study aims to have an average performance on the ANN model.

In addressing the concern of overfitting in this study, several strategies were implemented. We applied regularization techniques, including L1 regularization in Lasso Regression and leveraging regularization parameters in the XGBoost algorithm, aiming to control model complexity. Cross-validation, particularly k-fold cross-validation, played a pivotal role during model training, ensuring robustness by evaluating models across various data partitions. The optimization of hyperparameters through techniques like GridSearchCV and the utilization of ensemble methods, such as the StackedRegressor, were employed to strike a balance between model complexity and predictive performance. These concerted efforts were undertaken to mitigate overfitting risks while maintaining the models' accuracy and generalizability.

## 4. Results

A total of seven different models were trained and evaluated using the database of steel plate shear walls (SPSW). These models included Random Forest, Lasso, Gradient Boosting, XGBoost, AdaBoost, a Stacking Regressor, and an Artificial Neural Network (ANN). The performance of each model was assessed using various evaluation metrics, including mean squared error (MSE), R-squared score, and evaluation time. The following section presents a detailed comparison of the performance of each of these models in terms of their prediction accuracy and computational efficiency.

Regarding the R-squared score, the Artificial Neural Network (ANN) model outperformed all other models with an R-squared score of 0.918 on the validation dataset. The second-best performer was the XGBoost model, which had an R-squared score of 0.83. The Gradient Boosting model had an R-squared score of 0.701, followed by the Stacking Regressor with an R-squared score of 0.791. The Random Forest model had an R-squared score of 0.753, while the AdaBoost model had an R-squared score of 0.633. The lowest R-squared score was achieved by the Lasso model, with an R-squared score of 0.418. Overall, it can be seen that the ANN model had the highest prediction accuracy, followed by the XGBoost model. It is

also worth noting that the ANN model had the highest R-squared score for all data, training, validation and test set. The results are presented in Fig. 6. The results of the best-performed model are investigated separately in Fig. 7.

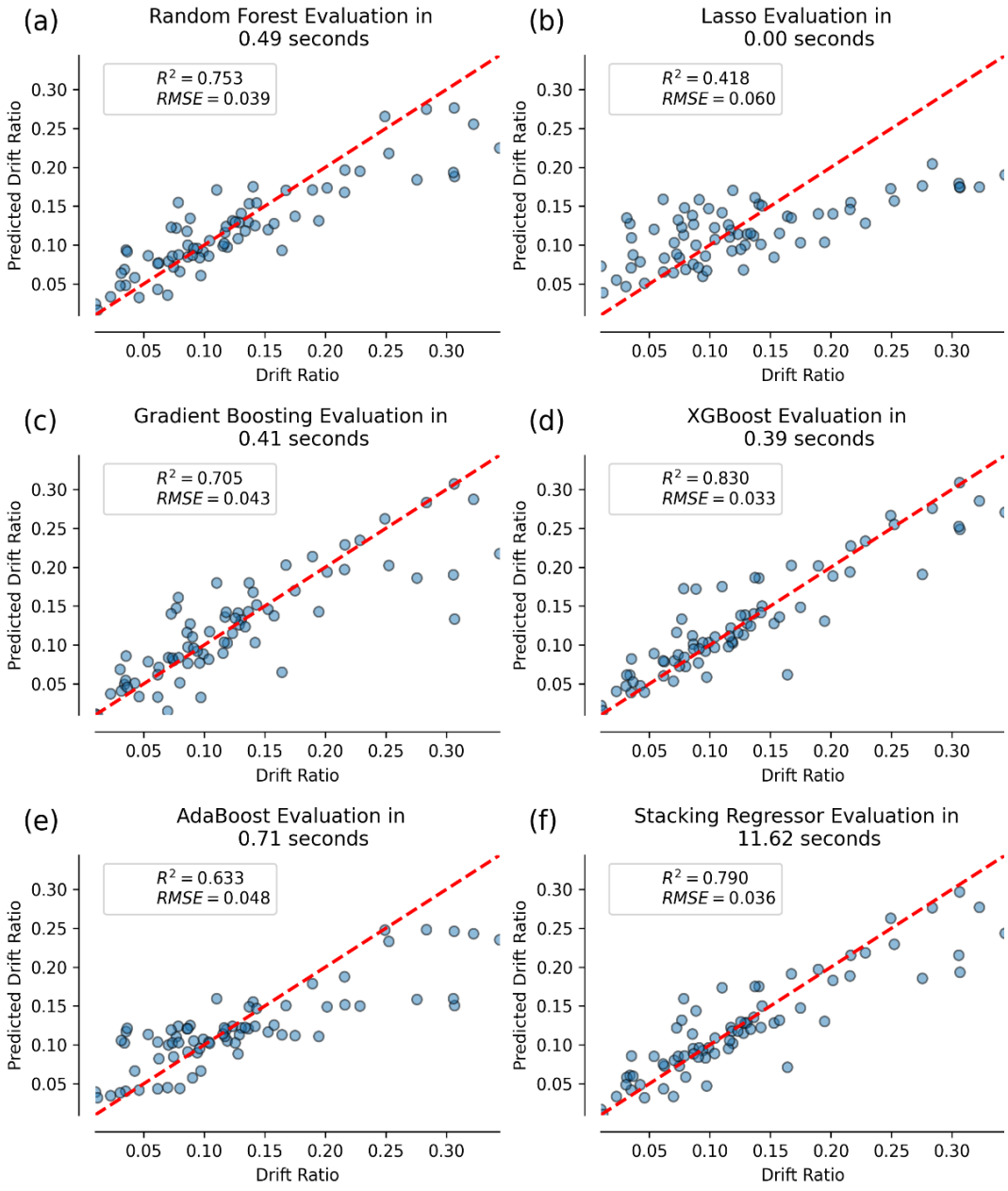



**Fig. 6.** Comparison of different models.

In terms of evaluation time, the best models were Lasso and XGBoost, with evaluation times of 0.01 and 0.4 seconds, respectively. These models could predict the ultimate drift ratio of SPSW systems with relatively low evaluation times. The Random Forest model, which had an evaluation time of 0.52 seconds, also performed relatively well in terms of evaluation time. On the other hand, the Gradient Boosting, Adaboost and Stacking regressor models had longer evaluation times of 0.43, 0.72 and 11.98 seconds, respectively. The Artificial Neural Network (ANN) model had one of the longest evaluation times of all the models, with an average evaluation time of 6.7 seconds. This is because the ANN model is a complex algorithm that requires more computational resources to train and evaluate. In addition, the loop in the script tests 1 to 60 hidden layers. However, the ANN model has the highest R-squared score of 0.9469 among the models. The evaluation time should be very low with a specific hidden layer size. It should be noted that the evaluation time of a model is an essential consideration in practice, as it can have a significant impact on the feasibility and cost of using the model in real-world applications.

When Fig. 7(a) is examined, the R-squared value of the ANN model of the training data was approximately 0.97, which is relatively high. Such a high R-squared number may indicate an overfitting issue. When an Artificial Neural Network (ANN) is overfitted, it underperforms on unknown, new data because the model was trained too well on the training data. Overfitting may occur due to noise, the small size of the training set, and the

complexity of the classifiers (Ying 2019). When a model has more parameters than training data, overfitting can happen, which prevents the model from generalizing to new data. However, the $R^2$ value of the validation data is 0.92 and the test data is 0.85, as shown in Figs. 7(b) and 7(c). These are also high $R^2$ values when viewed within the context of regression analysis, yet they are acceptable levels. A very high $R^2$ value of 0.95 was found in the regression analysis conducted on the entire data set in Fig. 7(c), similar to the other results.

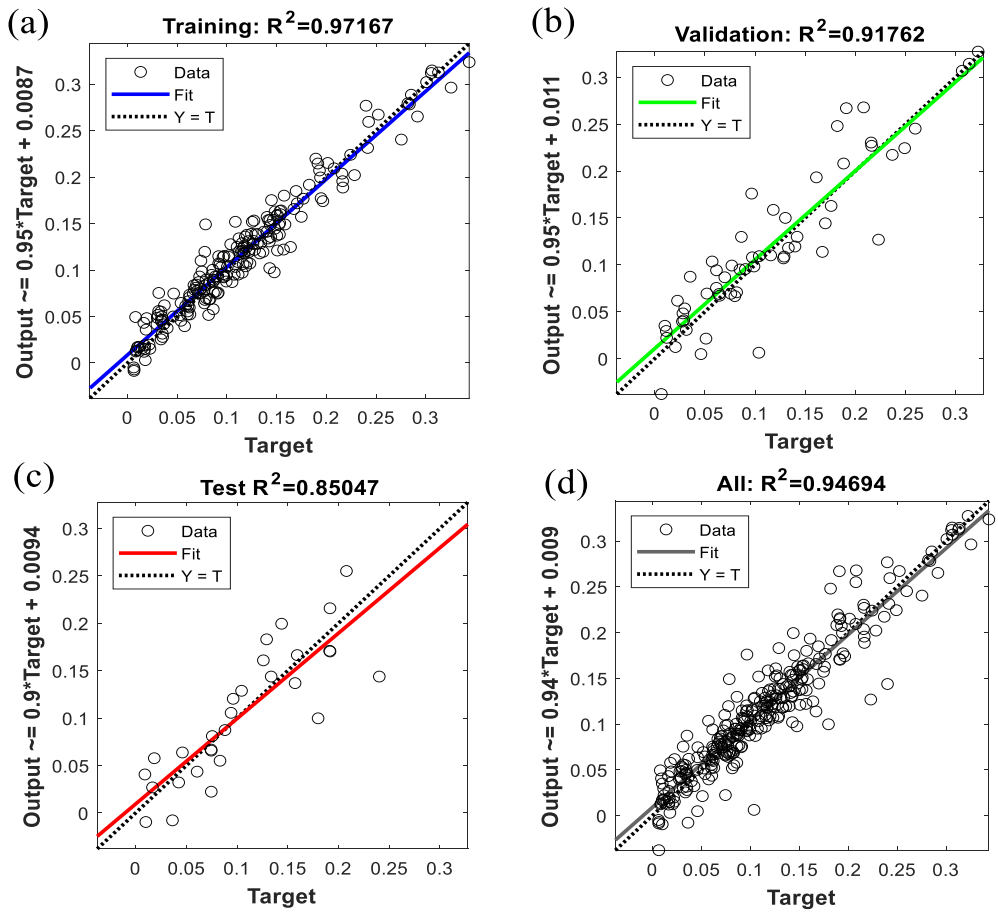



**Fig. 7.** ANN results: (a) Training data (70% of all data); (b) Validation data (20% of all data); (c) Test data (10% of all data); (d) All data predicted and target values.

Considering all these findings, XGBoost is among the most practical models regarding evaluation time and R-squared outcomes. Except for ANN, only XGBoost has a better R-squared value than the stacked regressor. The influence of the parameters on the results was investigated using various techniques through the XGboost model results.

The relative importance of each feature in a dataset can be evaluated using the machine learning technique known as feature importance analysis. First, a feature importance analysis was conducted and the results are presented in Fig. 8. In order to improve the performance of the model or to comprehend the underlying relationships in the data, it helps to identify the attributes that have the most influence on the model's outcome. In feature importance analysis, a tree-based model, such as XGBoost, was utilized to gauge the significance of various parameters in predicting the ultimate drift ratio of steel plate shear wall (SPSW) systems. The feature importance was calculated primarily through the evaluation of the Gini impurity or related metrics within these models. Higher feature importance scores indicate a stronger influence of the respective parameters on the predictive accuracy of the models.

When Fig. 8 is analyzed, it is obvious that $\omega_i$, $t_w$, and $P/P_{cr}$ are the most critical factors in the drift behavior of SPSW systems. According to the findings of the former study (Gürbüz and Kazaz 2022a, 2022b), these parameters directly impact the behavior of the system. Fig. 9 shows the relationship between the most essential three parameters with the ultimate drift ratio ($DR_u$).

The ultimate drift ratio and $\omega_i$ have a relatively high relationship, as can be shown in Fig. 9. A higher value of the column flexibility parameter, $\omega_i$, indicates that the columns are more flexible than the lower. It is expected that for high $\omega_i$ values, the ultimate drift ratio will take smaller values. Assessing the models with the lowest ultimate drift ratio reveals that they typically have thick plates and are subject to significant axial forces. This is another anticipated outcome that the SPSW system working principle can explain. Therefore, as the axial load on the columns and thickness of the plate increases, the columns are more proponed to buckle. Thick plates cause an increase in the inward force acting on the columns. Clearly, the

models with the lowest axial force also have the maximum capacity, as would be predicted. Another significant consequence of this figure is that very high drift values can be attained if the column axial load and plate thickness could be adjusted with the selected column section. Similar to the previous instance, the drift ratio capacity may be pretty low even when the column axial load and plate thickness are at their lowest levels in a poorly built model.

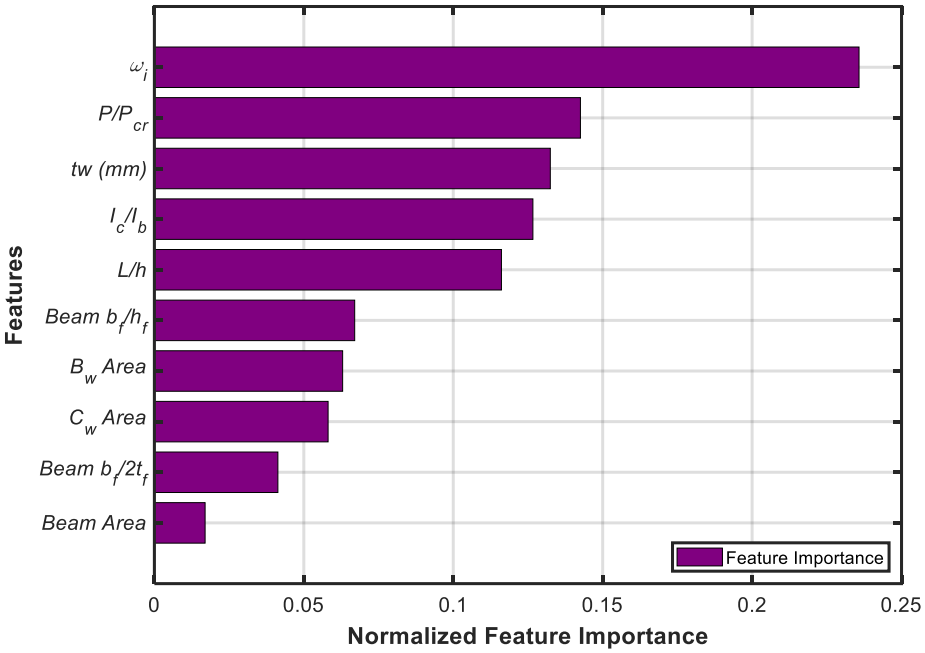



**Fig. 8.** Feature importance of XGBoost model.

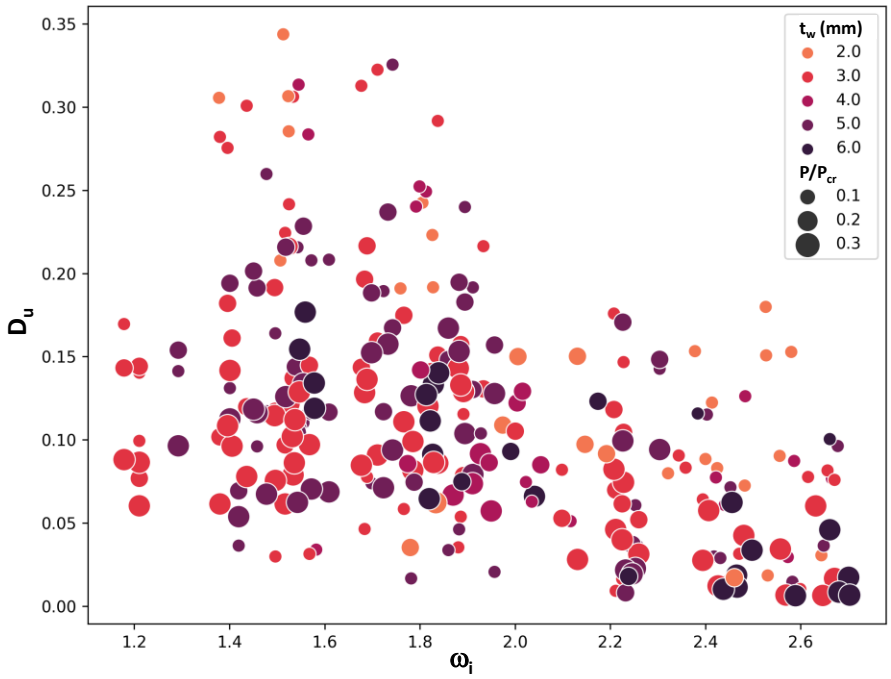



**Fig. 9.** Ultimate drift ratio – most effective parameters comparison.

Identifying the correlation between parameters is crucial to improve the data utilized in machine learning. Correlation between the input parameters is undesirable for machine learning. Because there is a high correlation between the input parameters, the model performs worse because the data set contains unused pairs. Additionally, any highly correlated parameters can improve the model's performance, while others can worsen it. The correlation coefficients for several variables in a dataset may be shown in a correlation matrix table. The matrix shows how each potential pair of variables in a table correlates. The correlation matrix is a table with rows and columns, and each cell displays the correlation between any two variables. Fig. 10 presents the correlation matrix of the parameters used in this study.

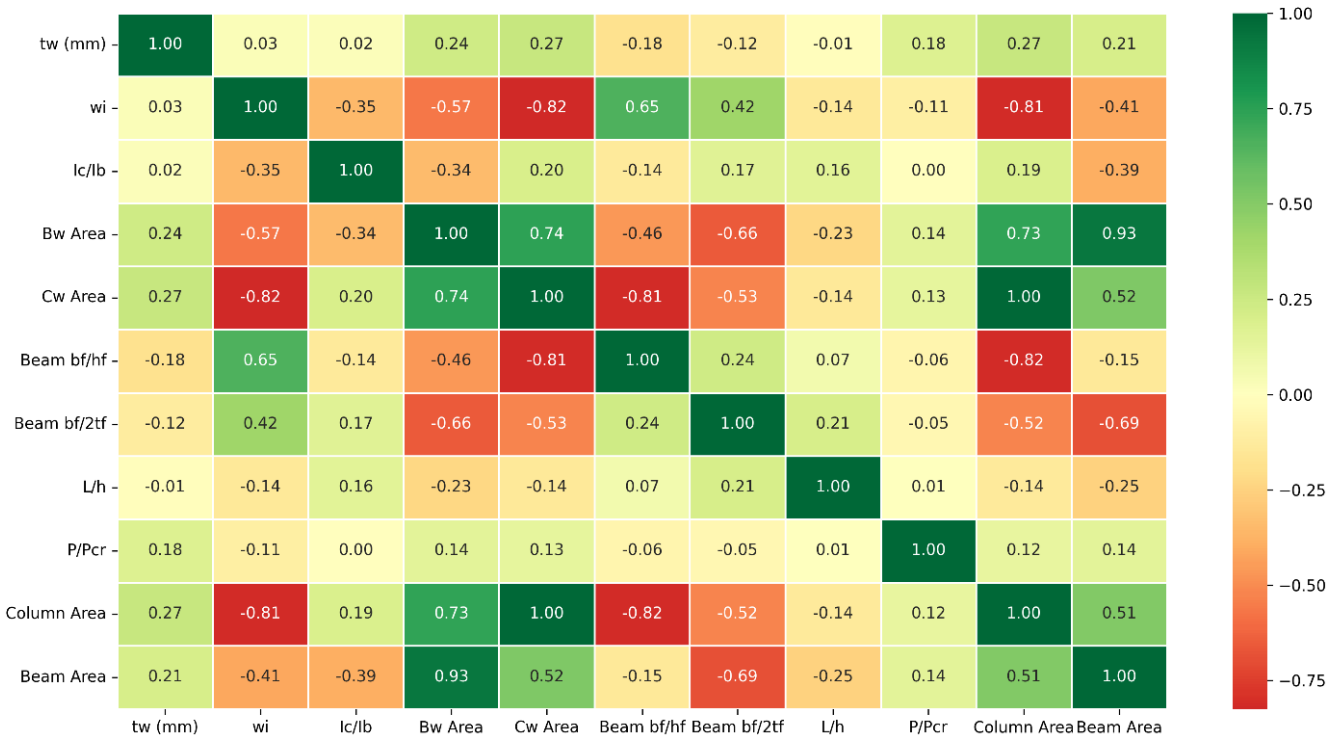



**Fig. 10.** Correlation matrix (XGBoost).

Some parameters have a very high correlation, as seen in Fig. 10. For instance, it is possible to conclude that the column area and the $\omega_i$ parameter have a strong inverse correlation. The relationship between beam web area and total beam area is also solid. The number of strong correlations might increase by looking at Fig. 10. That might be a reason for the relatively poor results of stacking-regressor. In this case, it can be concluded that the selected parameters should be optimized in machine learning.

On the other hand, the fact that the parameters $t_w$ and $P/P_{cr}$ typically do not exhibit a substantial correlation with any parameter indicates that two of the parameters which have the most extensive influence on the ultimate drift ratio were rather well selected. Similar to this, there was a low correlation between other factors and other relatively effective parameters, $L/h$ and $I_c/I_b$. Assuming that (except for $\omega_i$) parameters with high correlation are typically not very effective on the ultimate drift ratio would be reasonable.

In our analysis of feature importance, we utilized tree-based model, such as XGBoost, to gauge the significance of various parameters in predicting the ultimate drift ratio of steel plate shear wall (SPSW) systems. The feature importance was calculated primarily through the evaluation of the Gini impurity or related metrics within these models. Higher feature importance scores indicate a stronger influence of the respective parameters on the predictive accuracy of the models. Notably, parameters like $\omega_i$, $t_w$, and $P/P_{cr}$ emerged as the most critical factors affecting the ultimate drift ratio. These findings align coherently with established engineering knowledge in structural mechanics. For instance, $\omega_i$, the column flexibility parameter, directly correlates with the stiffness of columns, impacting the structural deformation capacity. Similarly, the plate thickness ($t_w$) and the ratio of applied axial load to critical load ($P/P_{cr}$) are indicative of the structural robustness and load-carrying capabilities of SPSW systems. These parameters' prominence in our analysis underscores their pivotal roles in governing the behavior of such systems, affirming their relevance from an engineering standpoint.

## 5. Conclusions

Predicting the ultimate drift ratio in SPSW systems is important for engineers and the construction industry. It plays a key role in ensuring structural resilience during seismic events, guiding design optimization, and enhancing overall safety and cost-effectiveness in construction projects.

The performances of various machine learning methods in predicting the SPSW ultimate drift ratio were comprehensively evaluated in this study. The models were compared and evaluated based on their R-squared value, root-mean-squared error, and evaluation time. A grid search method was used to fine-tune the models' hyperparameters in order to maximize their performance. A StackedRegressor class was also used to combine the predictions of the different models in an attempt to enhance the models' overall performance. Then, compared to the individual models, these combined results were used to create more accurate predictions.

Machine learning procedures employing feature importance analysis are very useful in determining the relative importance of each parameter in a dataset in order to improve the performance of the prediction model or to reveal the underlying relationships in the data. It was demonstrated that $\omega_i$, $t_w$, and $P/P_{cr}$ are the most critical

parameters in the deformation capacity of SPSW systems. These parameters directly impact the behavior of the system.

It was found that the Artificial Neural Network (ANN) model exhibited the most accurate predictions among the various machine learning algorithms evaluated. For the ANN model, the lowest R-squared value that was obtained for the training, validation, and test groups was 0.85. The R-squared value for the analysis conducted on the entire dataset was found to be 0.94, indicating a high level of accuracy. Among the other models, the XGBoost model had the second-highest accuracy level with an R-squared value of 0.93. However, the ANN model had one of the worst performances in terms of evaluation time due to the use of a loop to optimize the number of neurons in the algorithm. The optimization of the number of neurons has a direct impact on the results. The data used in the analysis, which consisted of 292 data points, did not account for very complex data and it can be inferred that the models would work well with much more complex data in real-life applications. The ability of the model to make profound contributions to the usability of this model is an important aspect to consider.

Several limitations should be acknowledged in this study, notably the dataset's size and complexity. The dataset used for model training and evaluation comprised 292 instances, which, while informative, might not encapsulate the full breadth of potential scenarios encountered in real-world applications. The relatively small dataset size could potentially limit the model's ability to generalize well to diverse or unseen data patterns. Moreover, the inherent complexity of the structural behavior of steel plate shear walls (SPSW) systems adds another layer of challenge, as the dataset might not comprehensively cover all possible variations and scenarios observed in practical applications. These limitations may influence the generalizability of the results, suggesting that caution should be exercised when applying these models to different or more complex real-world scenarios.

***Acknowledgements***

*None declared.*

***Funding***

*The authors received no financial support for the research, authorship, and/or publication of this manuscript.*

***Conflict of Interest***

*The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this manuscript.*

***Author Contributions***

*All of the authors made substantial contributions to conception and design, or acquisition of data, or analysis and interpretation of data; were involved in drafting the manuscript or revising it critically for important intellectual content; and gave final approval of the version to be published.*

***Data Availability***

*The datasets created and/or analyzed during the current study are not publicly available, but are available from the corresponding author upon reasonable request.*

## REFERENCES

Adeli H, Yeh C (1989). Perceptron learning in engineering design model. *Microcomputers in Civil Engineering*, 4, 247–256.

ANSYS (2016). Academic Research Mechanical, Release 17.2, Help System; *ANSYS. Inc.: Canonsburg, PA, USA.*

Breiman L (2001). Random forests. *Machine Learning*, 45(1), 5–32.

Chen T, Guestrin C (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-Augu, 785–794.

Deep Learning Toolbox - MATLAB. (n.d.). Retrieved January 11, 2023, from https://www.mathworks.com/products/deep-learning.html

Freund Y, Schapire R. E (1997). A Decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 119–139.

Gürbüz M, Kazaz İ (2022a). A numerical investigation on the limitations of design equations for steel plate shear walls. *Teknik Dergi*, 33(5), 12677–12708.

Gürbüz M, Kazaz İ (2022b). Numerical evaluation on the steel plate shear wall design parameters for improved cyclic behavior. *International Journal of Steel Structures*, 22(2), 409–429.

Hajela P, Berke L (1991). Neurobiological computational models in structural analysis and design. *Computers and Structures*, 41(4), 657–667.

Inel M (2007). Modeling ultimate deformation capacity of RC columns using artificial neural networks. *Engineering Structures*, 29(3), 329–335.

Kalman Šipoš T, Sigmund V, Hadzima-Nyarko M (2013). Earthquake performance of infilled frames using neural networks and experimental database. *Engineering Structures*, 51, 113–127.

Li CH, Tsai KC (2008). Experimental responses of four 2-story narrow steel plate shear walls. *Proceedings of the 2008 Structures Congress*, vol. 314, Vancouver, Canada.

Luo H, Paal SG (2019). A locally weighted machine learning model for generalized prediction of drift capacity in seismic vulnerability assessments. *Computer-Aided Civil and Infrastructure Engineering*, 34(11), 935–950.

Mahfuz Ud Darain K, Shamshirband S, Jumaat MZ, Obaydullah M (2015). Adaptive neuro fuzzy prediction of deflection and cracking behavior of NSM strengthened RC beams. *Construction and Building Materials*, 98, 276–285.

Nguyen HD, Dao ND, Shin M (2021). Prediction of seismic drift responses of planar steel moment frames using artificial neural network and extreme gradient boosting. *Engineering Structures*, 242, 112518.

Scikit-learn: machine learning in Python – scikit-learn 1.2.0 documentation. (n.d.). Retrieved January 10, 2023, from https://scikit-learn.org/stable/index.html

Sun H, Burton HV, Huang H (2021). Machine learning applications for building structural design and performance assessment: State-of-the-art review. *Journal of Building Engineering*, 33, 101816.

Thai HT (2022). Machine learning for structural engineering: A state-of-the-art review. *Structures*, 38, 448–491.

Tibshiranit R (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), 267–288.

Wu X, Ghaboussi J, Garrett JH (1992). Use of neural networks in detection of structural damage. *Computers and Structures*, 42(4), 649–659.

Xie Y, Ebad Sichani M, Padgett JE, DesRoches R (2020). The promise of implementing machine learning in earthquake engineering: A state-of-the-art review. *Earthquake Spectra*, 36(4), 1769–1801.

Ying X (2019). An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168(2).