This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic

M.Z. Naser, PhD, PE
School of Civil & Environmental Engineering and Earth Sciences (SCEEES), Clemson University, USA
Artificial Intelligence Research Institute for Science and Engineering (AIRISE), Clemson University, USA
E-mail: mznaser@clemson.edu, Website: www.mznaser.com

Abstract

The traditional approach to formulating building codes is often slow, labor-intensive, and may struggle to keep pace with the rapid evolution of technology and domain findings. Overcoming such challenges necessitates a methodology that streamlines the modernization of codal provisions. This paper proposes a machine learning (ML) approach to append a variety of codal provisions, including those of empirical, statistical, and theoretical nature. In this approach, a codal provision (i.e., equation) is analyzed to trace its properties (e.g., engineering intuition and causal logic). Then a ML model is tailored to preserve the same properties and satisfy a collection of similarity and performance measures until declared equivalent to the provision at hand. The resulting ML model harnesses the predictive capabilities of ML while arriving at predictions similar to the codal provision used to train the ML model, and hence, it becomes possible to adopt in line with the codal expression. This approach has been successfully examined on seven structural engineering phenomena contained within various building codes, including those in North America and Australia. Our findings suggest that the proposed approach could lay the groundwork for implementing ML in the development of future building codes.

Keywords: Building codes; Machine learning; Prediction.

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

1.0 Introduction

Building codes are guiding documents that home rules and recommendations aimed to standardize civil engineering guidelines and processes pertaining to design, analysis, construction, etc. (ASCE 2016). Such codes remain the cornerstone of civil engineering, providing a framework to establish building occupants' safety, health, and general welfare.

The process of creating codal provisions begins with collecting and analyzing data, often gathered from past building failures, tests, simulations, and empirical/practical experiences gained from well-documented incidents. All such data is systematically archived, studied, and evaluated to understand and learn from it to avoid repeating past failures (Ellingwood 1994). The same is then formulated and incorporated into the building codes. For instance, the International Building Code (IBC) (IBC 2018) stipulates the minimum standards for various aspects, like structural integrity and fire safety. By complying with such codes, engineers can minimize the risk of building collapse, fire hazards, and other potential accidents. Thus, adherence to codal provisions is not merely voluntary but a legal obligation and ethical responsibility (Law 2021). The open literature shows various cases where non-compliance with building codes resulted in catastrophic loss of life and property (Windapo and Cattell 2010; Yakubu 2019).

Many codal guidelines are communicated via equations (formulas). These formulas display the key parameters and the relationship of how such parameters are tied to a specific phenomenon. For example, Eq. 1 shows that the moment capacity of compact W-shaped steel beams is a function of the plastic section modulus, Z, and the yield strength (f_y) of the structural steel. In this format, an engineer can simply predict and/or evaluate the expected moment capacity of a given beam via

Eq. 1. More importantly, an engineer may also causally infer that an increase in either (or both) parameters can increase the moment capacity of the beam (and vice versa).

$$M = f_y \times Z \tag{1}$$

While some formulas stem from the deterministic nature rooted in physics and engineering principles, others are probabilistic, empirical, or arrived at via statistical methods (Lie 1992; Marasco et al. 2021). Despite the origin of codal provisions, variants of such provisions exist in building codes. Building committees have been noted to accept such variants as without these committees, provisions would not have been accepted and included in building codes.

In recent years, machine learning (ML) has shown great success in many fields, including civil engineering. ML models can analyze vast datasets, identify complex patterns, and make accurate predictions. A look into the existing literature displays studies wherein predictions from ML are reported to exceed those obtained from national and international building codes (Aladsani et al. 2022; Cakiroglu et al. 2022; Degtyarev and Tsavdaridis 2022; Naser 2023b; Tarawneh et al. 2021; Yahiaoui et al. 2023). This begs the question of how we can leverage such models.

However, ML models differ significantly from codal provisions. At the moment, most ML models remain driven by the supplied data. For instance, where a building code might specify an equation to calculate wind loads based on the height and location of a building, a ML model would first be trained to use a database of historical wind data and building performance to predict the same phenomenon easily. While ML models may excel at making predictions based on the patterns in the training data that may indeed surpass predictions obtained from codal provisions, the same

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

models can struggle when confronted with scenarios or data that significantly deviate from their training set (Krueger et al. 2021).

Furthermore, the dynamic nature of ML models, which are often updated and changed as they learn from new data, contrasts sharply with the static nature of building codes (which are updated but at a much more infrequent rate). For instance, the specifics of the ML model (algorithmic architecture, hyperparameters, etc.) used to predict a specific phenomenon may change significantly over time as the model is fed with new data. This may present a potential issue for engineers who rely on consistency in their design processes (Paleyes et al. 2022).

Perhaps, most importantly, is this lack of transparency of some ML models. This opaqueness can pose difficulties when attempting to integrate ML models into building codes, as these codes need to have clearly defined and understandable criteria so that engineers can follow and verify compliance (Naser and Ross 2022). While the data-driven nature of ML is its strength, it is fundamentally elemental not to overlook the value of engineering principles, intuition, and domain knowledge – especially in our field where decisions can impact occupants' safety (Naser 2021b).

Understandably, engineers may remain skeptical of ML models due to the above challenges and the absence of a regulatory framework that allows such models to be included in building codes. This absence of a framework means there are no defined standards to follow. This lack of standards makes it difficult for engineers to fully trust and adopt ML models. For example, engineers might be wary of using an ML model if there is not a universally accepted standard for such ML models. Given the absence of a structured way to integrate ML models into the design and construction

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

processes outlined by the building codes, engineers often opt for traditional, code-approved methods – even in scenarios requiring performance-based principles or out-of-the-box solutions.

On a more positive note, the rapid advancements in ML have now allowed the integration of eXplainable AI (XAI), both of which can help create more reliable and interpretable models that could be adopted into building codes. These two concepts refer to methods and techniques that allow users (i.e., engineers) to understand ML models' decision-making processes (Emmert-Streib et al. 2020). For instance, if a ML model could explain that it predicts a specific outcome (e.g., sectional capacity) based on specific characteristics of the building design (such as material properties and geometric features), engineers would be more confident in integrating these insights into their work than if the ML was a traditional blackbox model. Simply, if a ML model can explain its decisions in a way that aligns with the stipulations of codal provisions, this could significantly enhance the potential for ML's integration into these codes.

Providing transparency and a deeper understanding of the underlying relationships in the data can bridge the gap between the dynamic, data-driven world of ML and the static, more calm nature of building codes. From this lens, this paper proposes such a methodology. In this methodology, the first stage analyzes a codal equation to trace its properties (e.g., logic). Then, a ML model is tailored to learn and map the same logic – thereby transforming into a higher-order model that preserves the same properties of the codal equation. Finally, the ML model is tuned on big data to achieve superior performance, given the ability of ML to accommodate highly nonlinear relations. Our findings indicate that the proposed approach could set the stage for ML implementation in future building codes. While significant research and development are needed to refine these

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

techniques and demonstrate their reliability and effectiveness in the context of civil engineering and the involvement of regulatory and code-developing organizations, this work and its findings hope to jump-start a serious conversation toward such an effort.

2.0 A look into codal provisions

Codal provisions often undergo a series of iterations and must bypass rigorous discussions/voting by regulatory bodies, committee members, and the general public. Such provisions are commonly developed from first principles (i.e., mechanics). In the event that first principles are absent or immature, provisions can also be derived through statistical treatment of the outcome of physical tests or, to a lesser extent, numerical simulations.

In physical tests, a series of specimens are designed with preset features identified by various means (including empirically) and tested to generate an understanding that ties the examined features to the phenomenon at hand (i.e., flexural response). The outcome of such an investigation is then analyzed via statistical regression. To a certain extent, civil engineers favor using linear and nonlinear regression (Benjamin and Cornell 2014). Both forms of regressions home a number of assumptions that need to satisfy the data at hand as well as those pertaining to pre-selected parametric forms (e.g., the use of linear regression implies the validity of linear and additive nature of features, etc.) some of which may or may not be satisfied.

Evidently, a proper statistical regression analysis reduces to finding the best coefficients that can improve the accuracy of the pre-selected parametric form from the available tests/simulations data (Ribarič et al. 2008). Such an expression is further treated to attain the expected reliability measures inherently required by a building code's technicalities. Once such an expression is fully

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

verified, vetted, and voted upon, this expression is likely to be incorporated into a guiding document. This procedure has been duly noted in our literature, and one of the recent examples is that developed by Kuchma et al. (2019), credited with updating the one-way shear design provisions for reinforced concrete in the latest ACI 318 code adopted by the American Concrete Institute.

Depending on the comprehensiveness of a testing campaign, it is quite likely that additional tests or numerical simulations can be added to supplement the data at hand. For example, Franssen and Dotreppe (2003) followed a similar approach when developing the basis for Eurocode 2's assessment method for the fire resistance of columns. Similarly, Wade et al. (1997) combined tests with numerical simulations to derive an equation to evaluate the fire resistance of reinforced concrete columns for the Australian building code (AS3600). Other examples can also be found elsewhere (Feldman and Bartlett 2005; Ollgaard et al. 1971).

This work argues that, with the presence and/or absence of first principles, and when the primary motivation behind utilizing statistical-based expressions in building codes is their capacity to best fit existing data, then it is of merit to utilize higher-order models (such as those that fall under tree-based and neural network models, etc.) also capable of best fitting the data in a more accurate manner than that provided by the traditional counterparts. Noting how such models can better handle high dimensional data, has non-parametric freedom, and less adherence to data-related assumptions positions this method for creating suitable codal provisions.

A supporting philosophical view is that, for the same data and features, the statistical-based procedure of iteratively fitting a codal expression by best-fitting the coefficients of its predefined form can be viewed similarly to training a ML model to learn patterns in the available data (see Fig. 1). A primary difference is that the data at hand tune the coefficients for the already-defined parametric form of the former and the algorithmic logic (internal model structure) for the latter. One can then view the parametric form to be equivalent to the algorithmic logic since both tie the key parameters into arriving at the predictions. Yet, a critical difference between the above two approaches is that we are familiar with the procedure of statistical regression and, hence are able to trace the steps carried out to interpret the outcome of such analysis. As will be seen in Sec. 3, recent advances in explainability have now made this possible for ML (Molnar 2019).

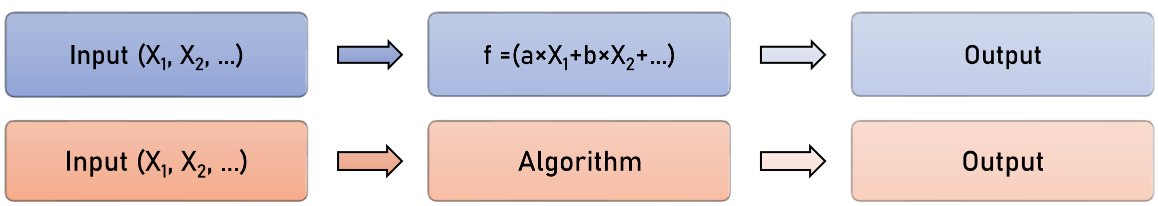


Fig. 1 A simplified illustration depicting two approaches for prediction (please note that additional steps could be involved (i.e., training, fitting, etc.) in a prediction problem).

3.0 Proposed methodology

3.1 General description

Suppose the same logic and properties of an empirical codal provision (i.e., equation) were imposed on a superior high dimensional ML model. In this case, such a model can be considered a natural substitute for the statistically-derived equation, as the model can now arrive at predictions and interpret their outcomes similar to the codal equation.

This becomes of importance as, in reality, while many of the phenomena tackled by civil engineers may not fully conform to the assumptions inherent in statistical methods used to derive empirical expressions, engineers still apply domain knowledge to assign the position and interaction between the involved parameters fitted into such expressions. Thus, the proposed methodology starts by constraining the algorithmic structure of a candidate ML model to one that imitates the same characteristics (in terms of engineering logic, domain knowledge, etc.) and the essence of the codal provision at hand.

This process is defined herein as *model equivalence*. While there could be multiple ways to establish equivalence (as will be detailed in Sec. 5.0), this work establishes equivalence by leveraging two eXplainable artificial intelligence methods, namely partial dependence plots (PDP) and accumulated local effects (ALEs) to constrain the algorithmic structure to yield PDPs/ALEs similar to those obtained from the codal equation (to a pre-specified degree). Such a degree is outlined via several similarity measures, five of which will be presented below. Finally, and once equivalence is established, model predictions are compared against those obtained from the codal equation via statistical measures. This section starts by covering the principles of PDP/ALE and then dives into the similarity measures selected to establish equivalence.

3.2 Equivalence through partial dependence plots (PDP)

The PDP explains ML models by depicting an individual (or more) feature's *marginal effect* on the prediction of a ML model while holding other features constant (Friedman 2001). The same plot can also be used to infer the type of relationship between each feature and the target. A PDP is created by varying the feature of interest across its range, and for each value, we make a prediction using the ML model or codal equation, holding all other features constant at their average values. The average prediction is then plotted against the values of the feature of interest

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

to show the marginal effect of that feature on the predicted outcome. Mathematically, the partial dependence of a model *f* with respect to a set of features X_S can be defined as:

$$PDP_{X_S} = \frac{1}{n}\sum_{i=1}^{n} f(X_S, X_{C_i}) \quad (2)$$

where X_{Ci} are the remaining features for the *i*-th observation, and *n* is the number of observations.

The key insight behind PDPs is that by averaging out the effects of all other features, then this tool can isolate the effect of the feature of interest. Thereby, the relationship between the target and a feature can be revealed. Fig. 2 illustrates a hypothetical PDP obtained from a ML model and from a physical equation. This PDP can show how changing the feature at hand while keeping all other features constant affects the predicted target. In this figure, one can see that the hypothetical model's PDP has a similar trend and behavior, to some extent, to that of the hypothetical equation. This implies that model and equation predictions are within certain a certain bound (i.e., percentage).

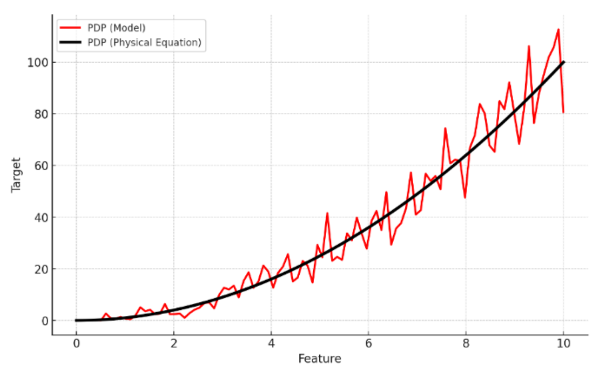


Fig. 2 A demonstration of PDPs [Note: the target and feature are assumed to be unitless for demonstration purposes.]

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

3.3 Equivalence through accumulated local effects (ALE)

While PDPs provide valuable insights, they can be biased when features are correlated because PDPs treat features independently. Thus, ALEs are another tool that can be used to explain ML models. However, unlike PDPs, ALEs provide a more accurate representation when features are correlated. For example, ALEs provide a way to visualize feature effects by taking into account the average effects of other features. Hence, ALE plots can be especially useful when there are interactions between features. ALEs handle interactions between features by only considering the local effects; hence they are not biased by global correlations like PDPs. They also are centered around the average prediction over the training data, which makes them easier to interpret in terms of the average prediction.

The ALE of a feature is calculated by accumulating its *local effects* (i.e., the difference between the prediction for a slightly higher value of the feature and the prediction for a slightly lower value for the same feature, averaged over the dataset). To calculate ALE, we first divide the feature of interest into intervals. For each interval, we compute the difference in the model's predictions at the upper and lower bounds of the interval while keeping other features constant. This difference is then accumulated across the intervals to obtain the ALE plot. Mathematically, the ALE for a feature x_i can be expressed as:

$$ALE_{x_i} = \frac{1}{n}\sum_{i=1}^{n}\left[f(x_{i,j+1}, X_{-i}) - f(x_{i,j}, X_{-i})\right] \quad (3)$$

Where $x_{i,j}$ is the value of feature *xi* for the *j*-th observation, and X_{-i} are the remaining features.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

The following example can come in handy to better compare PDPs and ALEs. For example, a PDP might show how the predicted sectional capacity changes with the cross-sectional area while averaging out the effect of the material strength. If there is a strong interaction between the cross-sectional area and the material (e.g., if changing the material significantly affects how the cross-sectional area influences the capacity), this interaction effect could be hidden in a PDP. On the other hand, suppose there is an interaction between the cross-sectional area and the material of the load-bearing member. An ALE plot can capture this interaction by showing the effect of changing the cross-sectional area on the predicted capacity, considering the material feature's average effect. For visualization purposes, both PDP and ALE are quite similar.

3.4 Detailed description of the proposed approach

The key challenge herein is that both PDP and ALE are only generated once model training is completed. In other words, these tools cannot be manipulated during the training process, and there is no direct way to train ML models to yield PDPs and ALEs that fit some criteria. Similarly, the traditional way of tuning model parameters (i.e., hyperparameters) to improve model accuracy cannot be applied herein to arrive at pre-specified PDPs and ALEs. Simply adjusting model parameters directly to match the PDPs and ALEs of the physical equation can be quite challenging because PDPs and ALEs are, as mentioned above, not directly connected to the model's parameters. Therefore, this challenge is inherently nontrivial and would involve creating a loss function that incorporates the discrepancy between the model's PDPs/ALEs and the target PDPs/ALEs and then optimizing this loss function to find the best parameters.

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

Thus, a new approach is proposed herein. This approach utilizes *learning by indirect feedback*. Rather than adjusting the model parameters directly to fit the PDPs and ALEs of the equation, we will be slightly adjusting the training data based on the discrepancy between the ML model's PDPs/ALEs and the equation's PDPs/ALEs and re-train the model on this adjusted data. Simply, we tune the ML model, and one of the means of tuning the model (among other standard methods often used in ML analyses) includes a step wherein the discrepancy between the PDPs/ALEs is minimized.

As will be demonstrated in the to-be-presented case studies, this approach has the potential to guide the ML model to learn the correct behavior (as defined by the PDPs and ALEs of the physical equation) while still being able to fit the actual data. This approach can transfer the embedded empirical scientific knowledge from codal provisions into ML and can be particularly useful in situations where we have a good understanding of the underlying physics or other processes, but the actual data is noisy or incomplete, so a purely data-driven model may not be accurate or reliable. In other words, while a traditionally-trained ML model can achieve high accuracy (as will be shown in the presented case studies), such a model does not generally attain PDPs and ALEs that match those from codal provisions. However, the proposed approach ensures that a ML will realize matching PDPs/ALEs and high predictive accuracy.

More specifically, the proposed approach entails the following steps:

1. Training an initial ML model and then calculating the PDP and ALE for each feature.
2. Then, computing the differences between the PDPs and ALEs of the model and those of the equation an engineer want the model to match.

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

3. The calculated differences are then used to adjust the target variable by iterating over each unique value in a feature and adjusting the target variable for all instances with a feature value in a certain interval.
4. The adjustment is calculated as the product of the learning rate (calculated using a decay formula = $learning\ rate/(1 + decay\ rate \times iteration)$, the difference in PDPs or ALEs, and a sensitivity parameter that represents how sensitive the target is to changes in this feature.
   - More specifically, The adjustment is calculated as follows:

$$adjustment_{pdp} = learning\ rate_{pdp} \times pdp\ difference[i] \times sensitivity_{pdp}$$

$$adjustment_{ale} = learning\ rate_{ale} \times ale\ difference[i] \times sensitivity_{ale}$$

   - Here, $Learning\ rate_{pdp}$ and $learning\ rate_{ale}$ are the learning rates for the PDP and ALE, respectively, calculated using a decay formula, which measures how quickly the model should adjust its predictions based on the differences. *pdp difference[i]* and *ale difference[i]* represent the absolute difference between the model's PDP/ALE and the equation's PDP/ALE for the current feature value. $sensitivity_{pdp}$ and $sensitivity_{ale}$ are the sensitivities of the output to the feature for PDP and ALE, respectively. These measure how much the output changes with respect to a small change in the feature. The adjustments are then applied to the target values ('FR') in the training data where the feature value is between the current value and the next value.

5. This adjustment is then added to the target variable for all instances where the feature value falls within a certain range.
6. The process is repeated until the maximum number of iterations (default is set to 100) is satisfied or until the differences between the model's PDPs and ALEs and the equation's PDPs and ALEs fall below a certain similarity threshold as defined per each measure.
    - If the model's PDPs/ALEs are similar to the equation effect's PDPs/ALEs (as per the predefined similarity measures), the process continues to the next feature. If not, the targets in the training data are adjusted based on the differences, and the model is re-trained.
    - If similarity is not achieved for all features within the maximum number of iterations, the algorithm moves on to the next measure.
7. After looping over all measures, a similarity check is examined to verify if the similarity was achieved for all features for all measures.
    - If so, each model that satisfies a specific measure is retained.
8. Finally, an ensemble model is created using all the retained models. The PDP and ALE for the ensemble model are calculated. This ensemble model, therefore, represents a combination of models, each trained for a different measure and adjusted for each feature, based on the differences between the model's PDPs/ALEs and the equation's PDPs/ALEs.

As one can see, there are three main loops, a similarity measures loop, an iteration loop, and a feature loop. Depending on the scenario at hand, an engineer can opt to choose if each loop needs to satisfy one/all measures, iterations and/or features. Similarly, the number of iterations may need

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

to be adjusted per problem/equation. After testing this approach for a number of problems, it became clear that a balance across the three loops is to be accomplished. In most cases, arriving at an ensemble with models satisfying 2-3 measures, all features, and within the maximum number of iterations seems to yield balanced results. Additional testing and verification are likely to continue in the future.

3.5 Similarity measures

As mentioned above, the relationships between features in building codes are governed by physical equations/provisions. Thus, if a ML model is developed to mimic these physical processes, then PDP and ALE of the model need to also mimic the logic of the physical equation. The elemental idea here is that if the PDP and/or ALE of a feature in the model follows the same trend as the physical equation, this implies that the ML model has learned the underlying engineering intuition and causal logic of the physical process. For example, if the physical equation describes a linear relationship between a feature and its target, and the PDP or ALE of these variables in the model also shows a linear relationship, this would suggest that the model has learned this relationship.

Now, we move to present the selected similarity measures. These measures were selected to be largely independent of each other to maximize their strengths and minimize their limitations. The derivation of these measures can be found in their respective works cited below.

3.5.1 Normalized Euclidean distance

The normalized Euclidean distance function calculates the Euclidean distance between two vectors and then normalizes this distance by the number of elements in the vectors (see Fig. 3) (Bernardin et al. 2017). Normalizing the distance by the number of elements is a way to make this measure scale independent. This can be useful when comparing distances across datasets with different numbers of elements. The Euclidean distance is the straight-line distance between two points in space and is calculated as:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + \cdots + (p_n - q_n)^2} \quad (4)$$

where p and q are two points in an n-dimensional space. This measure returns a zero if the shapes are identical and unity when completely dissimilar (the default setting was arbitrarily chosen as 0.25 in this study).

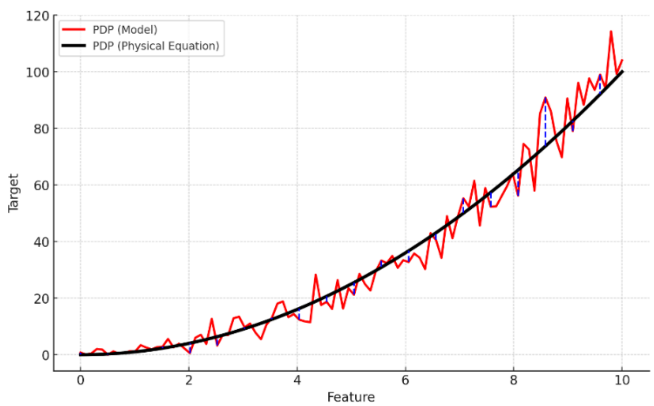


Fig. 3 Illustration of the normalized Euclidean distance measure (presented in dashes)

3.5.2 Fréchet distance

The Fréchet distance is another measure of similarity between two curves that takes into account the location and ordering of the points along the curves (Aronov et al. 2006). This measure is formally described as the length of the shortest leash required to let a pet and its owner walk along their separate paths, where the pet and the owner can vary their speed but cannot go backward (see Fig. 4). The Fréchet distance function implemented here calculates the discrete Fréchet distance between two curves, represented as arrays, and results in a single value (with the same units as the target value) representing the similarity between the two curves. This measure assumes that the curves are continuous and connected and that the order of points along the curve matters. Since this measure returns similarity in the same units as the target, a default setting was arbitrarily chosen as 10% of the observed target observation in this study).

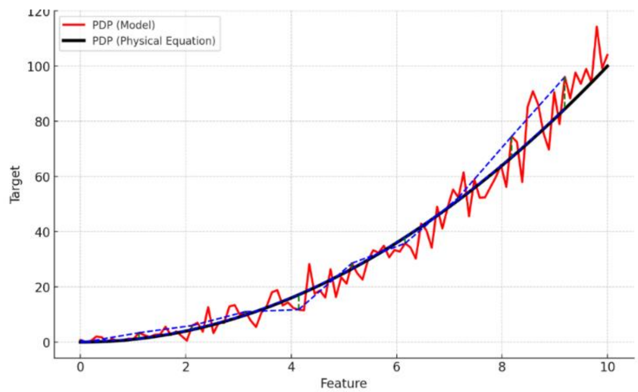


Fig. 4 Illustration of the Fréchet distance measure (presented in dashes)

3.5.3 Bounding measure

The custom bounding measure calculates an upper and a lower bound based on a specified threshold (see Fig. 5). The idea is to determine whether the PDP/ALE values from the ML model fall within a certain range of values defined by a physical equation. The measure returns the proportion of model values that fall within the bounds. This measure assumes that the physical equation provides a ground truth range of acceptable values. One should note that the choice of threshold is arbitrary (taken as 10% in this study) and that this measure does not take into account the order of the values.

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

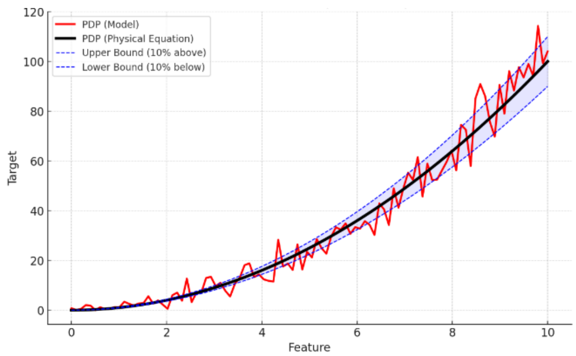


Fig. 5 Illustration of the bounding measure (presented in dashes)

3.5.4. Coefficient of determination and correlation coefficient

In addition to the above four measures, two commonly used statistical measures can be used to establish similarity between PDPs or ALEs of a model and a physical equation: the R^2 (coefficient of determination) and the correlation coefficient (r) are also used. The R^2 represents the proportion of the variance for a dependent variable that is explained by a/number of features in a model. The r, on the other hand, measures the strength and direction of a linear relationship between two variables. In the context of comparing PDPs or ALEs of a model and a physical equation, a high R^2 or r approaching unity would suggest a high degree of similarity. A similarity of more than 70% was deemed to be strong, as noted in (Smith 1986).

$$R^2 = 1 - \sum_{i=1}^{n}(P_i - A_i)^2 \,/\, \sum_{i=1}^{n}(A_i - A_{mean})^2 \quad (5)$$

$$R = \frac{\sum_{i=1}^{n}(A_i-\bar{A}_i)(P_i-\bar{P}_i)}{\sqrt{\sum_{i=1}^{n}(A_i-\bar{A}_i)^2\sum_{i=1}^{n}(P_i-\bar{P}_i)^2}} \quad (6)$$

where, A: actual measurements, P: predictions, n: number of data points.

3.6 Machine learning models

Five ML algorithms were used herein, and these were selected as they are commonly used algorithms in our domain, as noted in a recent survey (Tapeh and Naser 2022). All ML algorithms are used in their default settings (including hyperparameters) for transparency and to allow repetition of the proposed approach. The following sections briefly describe such algorithms, and their full description and scripts can be found in their respective references, as cited below. The specific script used in creating the provided analysis can be requested from the author.

3.6.1 Decision trees (DT)

A decision tree is a supervised ML algorithm that uses a tree-like graph of decisions and their possible consequences (Naser 2021a). It is a flowchart-like structure, where each internal node denotes a test on a feature, each branch represents an outcome, and each leaf node holds a class label. The process of constructing a decision tree involves selecting features to test at each node in the tree to minimize the total variance of the target variable among the instances at each node. There are several strategies for making this choice, but a common one is to use information gain, which is related to the concept of entropy (a measure of impurity) (Kent 1983).

3.6.2 Random Forest (RF)

Random Forest (RF) is a versatile ensemble learning algorithm that combines the power of decision trees and randomization techniques (Breiman 2001). The RF algorithm constructs an ensemble of decision trees by randomly selecting subsets of the training data and features. The individual trees are trained independently using the selected subsets, and their predictions are aggregated to produce the final prediction. Each decision tree in the RF ensemble is built using a

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

random subset of the training data (known as bootstrap aggregating or bagging). This random sampling process helps to reduce the risk of overfitting and improves the model's generalization ability. Additionally, the algorithm selects a random subset of features at each split in the tree construction, introducing diversity among the trees and reducing the impact of individual features.

3.6.3 Extreme gradient boosted trees (XGboost)

XGBoost is based on the gradient boosting framework (Scikit 2020; XGBoost Python Package 2020). Gradient boosting is a method that goes through cycles to iteratively add models into an ensemble. It begins by initializing an ensemble with a single model, whose predictions can be pretty naive. Then, the algorithm begins to add new models that aim to correct the errors of the combined ensemble. In this way, the models are added sequentially until no further improvements can be made. One of the main improvements that XGBoost brings for tree-like models is a more regularized model formalization to control overfitting, which tends to be a problem with gradient boosting.

3.6.4 Light gradient boosting machine (LGBM)

The Light Gradient Boosting Machine (LGBM) is an improvement over the XGboost as it combines the strengths of gradient boosting and the LightGBM framework to achieve high performance (Ke et al. 2017). This optimizes the objective function by iteratively fitting new trees to the negative gradient of the loss function with respect to the current predictions. The final prediction is obtained by summing the predictions of all the trees. The exclusive feature bundling technique in LGBM groups data points in a near-lossless way, leveraging similarities within the data. This technique improves computational efficiency by reducing the number of distinct feature values and enabling efficient memory usage during the training process.

3.6.5 Support Vector Machines (SVM)

The Support Vector Machines (SVM) algorithm aims to find the optimal hyperplane that maximally separates the data points (Scikit n.d.). The objective is to minimize the deviation or error between the actual data points and the hyperplane while considering the regularization parameters. To handle non-linearly separable data, SVM utilizes kernel functions to transform the input data into a higher-dimensional feature space where it becomes linearly separable.

3.7 Performance metrics

Once the ensemble is created. This ensemble is trained by randomly shuffling and training the data into training (T), validation (V), and testing (S) sets. The ensemble is trained and validated against the T and V sets and then examined on the S set following a k-fold cross-validation procedure, wherein the collected dataset was randomly split into test and training sets of $k = 10$ groups. The ensemble is trained using nine unique sets and validated on the tenth remaining set until each unique set is used as the validation set.

Then, in lieu of the above discussed measures selected to verify and compare the similarity between the PDPs/ALEs of the ML model and codal provision, the validity of the created ML models is also examined through another independent performance metric, namely, the Mean Absolute Error (MAE) as listed below, as well as R^2 (Naser et al. 2021; Tapeh and Naser 2022).

$$MAE = \frac{\sum_{i=1}^{n}|P_i - A_i|}{n} \tag{7}$$

4.0 Case Studies

Seven phenomena and their respective codal provisions are used herein to showcase and verify the proposed methodology. These are presented in the order of their complexity (i.e., the number of involved features). Two phenomena are taken from ACI 318 (Building Code Requirements for Structural Concrete and Commentary). Two phenomena were curated from ASCE29 (Standard Calculation Methods for Structural Fire Protection), wherein one phenomenon also appears in the Technical Report no. 10 by the National design specification (NDS) for wood construction. The fifth phenomenon is taken from the Australian Building Code (AS3600) for concrete construction. The sixth case study is taken from the American Institute of Steel Construction (AISC) Steel Construction manual. The final case study is present in multiple building codes as it deals with the elastic deformation of beams.

Each phenomenon was examined with data that satisfies the practical range of the codal provision at hand, and this data also satisfy the recommendations of the recent researchers aimed to establish the health of data (Van Smeden et al. (Smeden et al. 2018) with a minimum set of 10 observations per feature, Riley et al. (Riley et al. 2019) with a minimum set of 23 observations per feature, and Frank and Todeschini (Frank and Todeschini 1994) with a minimum ratio between the number of observations and features as 3 and 5). In addition, a random ML algorithm from those listed earlier was selected for each case study to create the tuned ensemble, and the other algorithms were used without tuning (i.e., without being incorporated into the proposed equivalence approach) for comparison. The rationale behind the decision is to show the proposed approach's wide applicability and maintain the page limitation of this work, as examining each individual ML model for each case study pushed the size of this paper beyond what is commonly expected. All case studies used the default settings listed above. In the upcoming PDP and ALE figures, the reader must remember that the vertical axis represents the partial and accumulated effects on the target instead of the target itself.

4.1 ACI318: Code Section 19.2.3.1 on modulus of rupture

In the context of building design and construction, the tensile strength of concrete is particularly important as it helps to determine the concrete's ability to resist tensile and torsional forces, as well as cracking (say due to environmental factors like temperature changes and freeze-thaw cycles, etc.). The tensile strength of concrete is generally only about 10% of its compressive strength and measures the force required to break/crack concrete.

Although the tensile strength of concrete increases with an increase in the compressive strength, the ratio of the tensile to the compressive strength decreases as the latter strength increases. The tensile strength is often approximated proportionally to the square root of the compressive strength of concrete. More specifically, ACI 318 Code Section 19.2.3.1 the modulus of rupture for normal strength concrete for use in calculating deflections as obtained from statistical analysis (see (McNeely and Stanley 1963; Raphael 1984)):

$$f_r = 7.5\sqrt{f_c'} \tag{8}$$

As one can see, this is a simple statistical equation and presents a suitable candidate for a first case study. Thus, the presented approach above will be applied to creating a ML ensemble that mimics this equation. Due to the simplicity of this problem, one similarity measure (normalized Euclidean

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

distance) was used. Upon the successful tuning of the ensemble, the $R^2_{ensemble}$ was unity and $MAE_{ensemble}$ was calculated at 0.192, and the same metric, when calculated for the RF (without tuning), was at 0.203, implying a 5.4% improvement in accuracy. The outcome of this approach is presented in Fig. 6. This figure shows that the PDP and ALE of the equivalent ensemble (made from RF) match well throughout the full series with the above codal equation. It is also worth noting that the PDPs and ALEs from all tree-based algorithms also matched those of the codal equation well. This can indicate that such models can predict this phenomenon without actually being tuned. This working hypothesis is being examined at the moment in a separate work and, at the time of this work, appears to be true for simple phenomena.

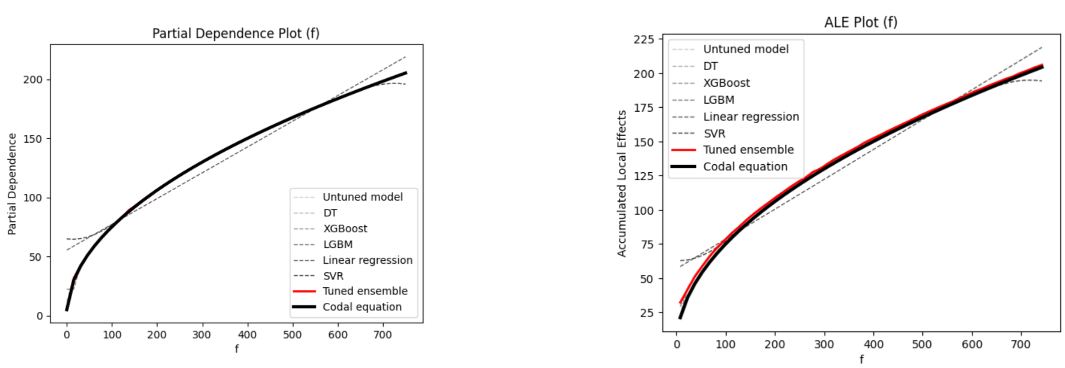


Fig. 6 Comparison for case study no. 1(in psi) [Note: *f* denotes the compressive strength of concrete. The base ML mode: RF.]

4.2 ASCE29: Standard Section 3.3 on the design of fire-resistive exposed wood members

ASCE 29 is a standard for calculating the fire resistance of structural members and barrier assemblies made from structural steel, un/reinforced concrete, timber, and masonry and is developed as a joint effort between ASCE's Structural Engineering Institute and the Society of Fire

Protection Engineers (SFPE). This standard contains a number of empirical codal provisions that can be used to evaluate/predict the fire resistance of structural members.

Sec. 3.3 in this standard provides guidance on calculating the fire resistance rating (or time to failure in minutes, *t*), measured in minutes, of timber beams with a least nominal dimension of 6 in. (or 140 mm). For the severe case of beams exposed on four sides, *t*, is calculated as:

$$t = \gamma z b[4 - (b/d)] \tag{9}$$

where, γ is a constant (2.54 min/in. or 0.1 min/mm), *z* is the load factor equals 1.3 for lightly loaded beams with a ratio < 50%, *b* and *d* are the actual breadth and depth of a beam. It is worth noting that the same empirical equation also appears in the Technical Report no. 10 by the National Design Specification (NDS) for wood construction (NDS 2018). As one can see, this empirical expression contains four parameters, two of which are considered constants and hence are treated as such, and hence only *b* and *d* are considered as features. This case study applied the proposed methodology by using LGBM as the base ML model.

In this case study, only three similarity measures converged with default settings (i.e., Fréchet distance, the Bounding measure, and the normalized Euclidean distance). Special tuning was not applied to converge the other measure as the realized results were deemed acceptable (see Fig. 7). The $MAE_{ensemble}$ was calculated at 0.859 min, and the same metric, when calculated for the same algorithm (without tuning), was at 1.111 min. R^2 between the ensemble and the codal equation was unity. It is clear that the tuned ensemble outperforms all individual models in matching the codal equation's PDPs and ALEs. In this case study, and as shown in the next study, matching PDPs is often seen to be faster (and affordable) than matching ALEs. This stems from the more complex

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

derivation nature of ALEs compared to PDPs. Given this observation, the proposed methodology was appended to allow the user/engineer to select if they seek to match PDP, ALE, or both. Such an amendment did not alter the analysis shown in the first case study.

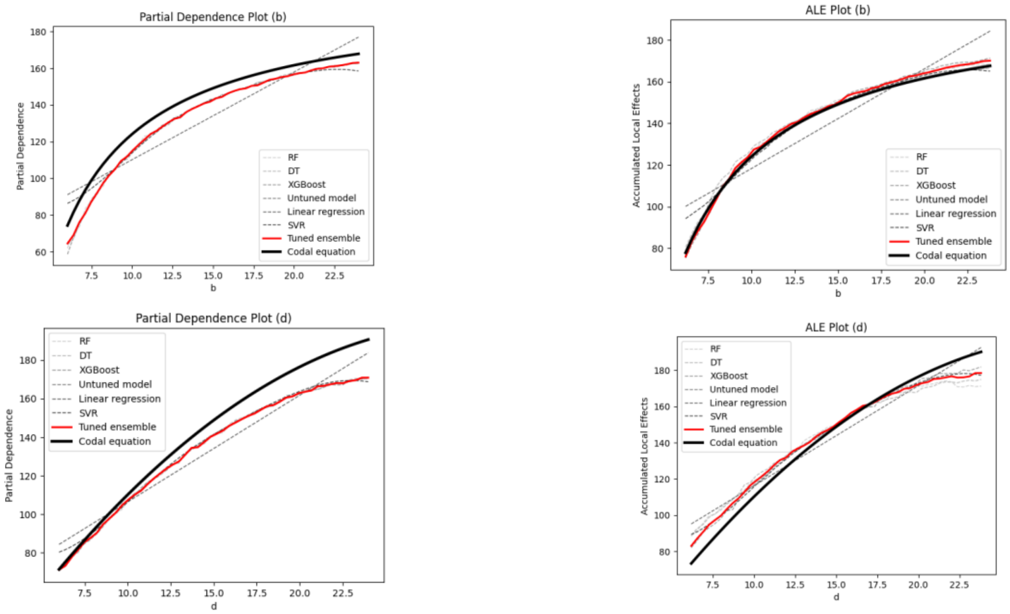


Fig. 7 Comparison for case study no. 2 (in min) [Note: *b* and *d* denote the breadth and depth of timber beams, respectively. The base ML mode: LGBM.]

4.3 ACI318: Table 22.5.5.1 on shear strength for beams without minimum shear reinforcement

ACI Code Table 22.5.5.1 lists the following equation for calculating the shear strength for beams without minimum shear reinforcement. This equation results from an iterative process led by the Joint ACI-ASCE Committee 445 and Subcommittee E of ACI Committee 318.

$$V_c = \left[8\lambda_s \lambda \rho^{1/3} \sqrt{f_c} + N_u/6A_g\right] bd \tag{10}$$

where ρ_w is the flexural reinforcement ratio for tension reinforcement in a section of $A_s/b_w d$, N_u is the member-factored axial load, and A_g is the gross member area. The lightweight-aggregate

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

concrete factor, λ, was defined previously. The value of V_c shall not be taken as less than zero, nor as greater than $\lambda 5\ (f_c')^{0.5}\ b_w d$, and $N_u/6A_g$ shall not be taken as greater than $0.05f_c'$. In this case study, beams with applied axial forces were not considered. In addition, λ_s (size effect term) and λ were taken as unity for simplicity.

The LGBM was used in this case study as the base model. Figure 8 shows the behaviour of the tuned ensemble in terms of PDPs and ALEs after satisfying two similarity measures, namely the Fréchet distance and the Bounding measure. The $MAE_{ensemble}$ was calculated at 3444.348 lbs and at 3620.0 lbs for the untuned model. The R^2 for the ensemble was 99%. Clear matching between the ensemble and codal equation is evident.

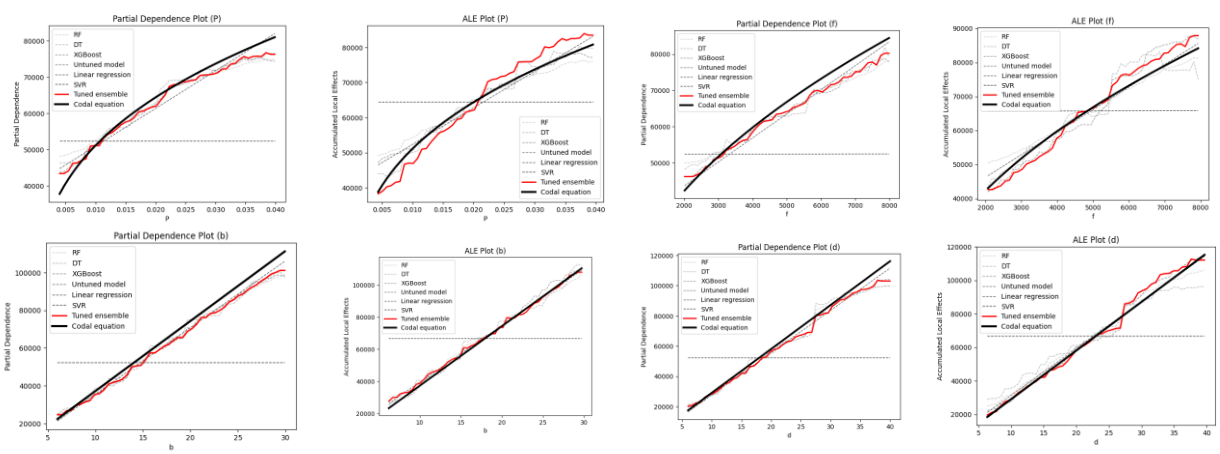


Fig. 8 Comparison for case study no. 3 (in lbs) [Note: The base ML mode: LGBM.]

To further examine the performance of the tuned ensemble (as well as the proposed approach), an additional layer of validation is followed herein. In this process, 100 random beams were selected from ACI's shear database (Reineck et al. 2003) for reinforced concrete members without shear reinforcement used in deriving the codal expression were examined against the codal expression and tuned ensemble. Figure 9 shows the results of this comparison and notes that the ensemble achieves a better predictive capability than the codal expression in predicting the shear strength of the selected beams ($MAE_{ensemble}$ = 4930.51 lbs, $MAE_{codalexpression}$ = 5219.73, $R^2_{ensemble}$ = 84.1% and $R^2_{codalexpression}$ = 80.83%). It is worth noting that this additional external verification was not possible to conduct in the other case studies as the data used to create the codal provisions of interest were not available.

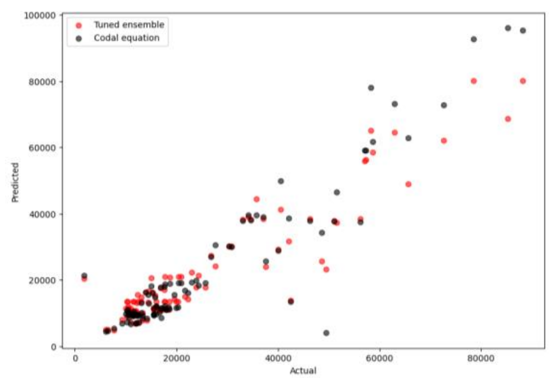


Fig. 9 Comparison against 100 randomly selected beams from ACI's shear database (in lbs) [Note: the database is accessible in (Reineck et al. 2003)]

4.4 ASCE29: Standard Section 5.2.3 on concrete-filled hollow steel columns

Another complex codal provision is tackled in this case study. The ASCE29 also presents guidance on evaluating the fire resistance rating of hollow steel columns filled with unreinforced normal weight concrete through the following provision. This equation was established empirically through a number of publications and has been verified by computer simulations at the National Research Council of Canada (NRC) (Kodur 1999; Kodur and MacKinnon 2000). It is worth noting that this expression is also listed under Sec. D-2.6.6. in the National Building Code of Canada and is also applicable to tubular shapes; however, only the case of circular tubes is presented herein.

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

$$R = a\frac{(f+20)}{60(L_e-1000)}D^2\sqrt{D/C} \tag{11}$$

where, R is the fire resistance rating in hours, a is 0.07 for circular columns filled with siliceous aggregate concrete, and 0.08 for circular columns filled with carbonate aggregate concrete. f is the specified 28-day compressive strength of concrete (MPa), L_e is the column effective length (mm), D is the outside diameter for circular columns (mm), and C is the compressive force due to the unfactored dead load and live load in kips (kN).

This codal provision is accompanied by additional stipulations that were enforced in this analysis, including, the required R rating shall be less than or equal to 2 hours, the specified f shall be equal to or greater than 20 MPa and shall not exceed 40 MPa. the column's effective length shall be at least 2000 mm and shall not exceed 4,000 mm, the outside diameter shall be at least 140 mm and shall not exceed 410 mm, and the compressive force shall not exceed the design strength of the concrete core determined in accordance with AISC LRFD-94, “Load and resistance factor design specification for structural steel buildings.”

The RF algorithm was used as the base model. The convergence of the ensemble on the PDPs and ALEs was difficult to obtain, and hence the presented results in Fig. 10 are shown for a converged run wherein only the PDPs were matched using all aforementioned similarity measures were satisfied. Despite only matching the PDPs, the ALEs seem to also attain some degree of matching resemblance to that from the equation. The $MAE_{ensemble}$ was calculated at 0.281 hours and at 0.471 hours for the untuned model – an improvement of about 40%.

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

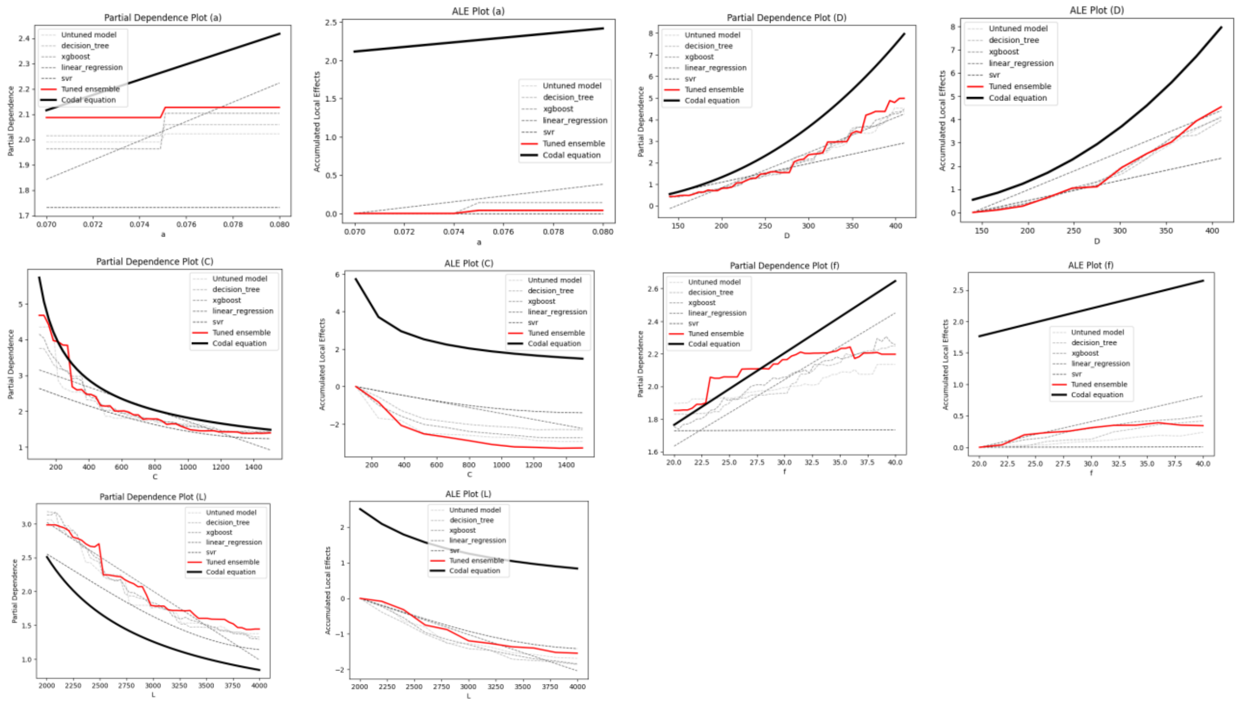


Fig. 10 Comparison for case study no. 4 (in hours) [Note: The base ML mode: LGBM.]

4.5 AS3600 Fire resistance of concrete columns

This case study tackles a more complex codal provision with six variables. A former edition of the Australian concrete code (AS3600) homed the following statistical-based codal equation to evaluate the fire resistance of concrete columns (Wade et al. 1997). This equation was fitted using the results of standard fire tests and numerical simulations.

$$R = \frac{k \times f^{1.3} \times B^{3.3} D^{1.8}}{10^5 \times N^{1.5} \times L^{0.9}} \tag{12}$$

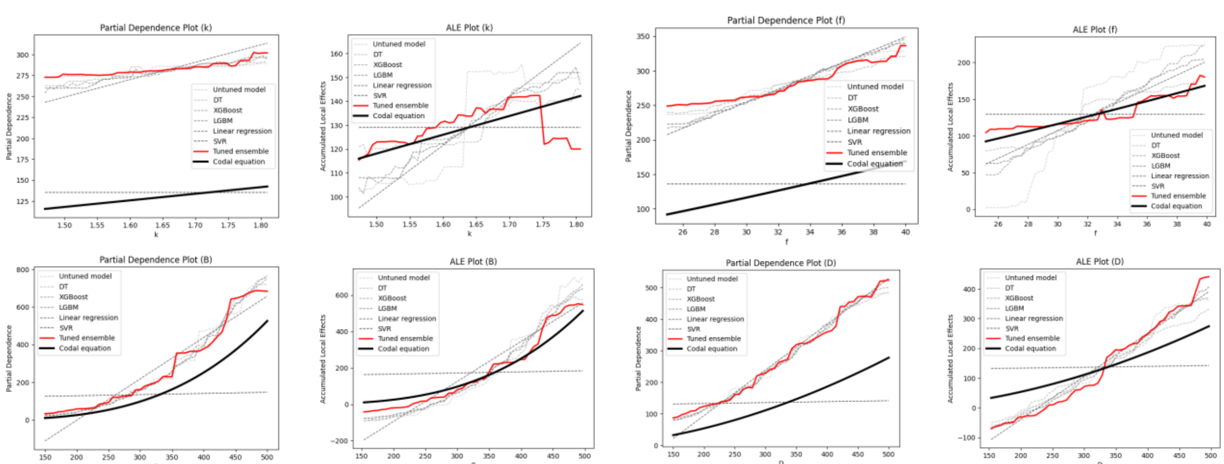
where, R is the fire resistance (min), k is the constant dependent on cover and steel reinforcement ratio (equals to 1.47 and 1.48 for a cover less than 35 mm and greater than or equal to 35 mm, respectively), f_c is the 28-day compressive strength of concrete (MPa), B is the least dimension of

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

column (mm), *D* is the greatest dimension of column (mm), *N* is the axial load during fire (kN), and *L* is the effective length (mm).

This case study utilizes the RF algorithm as a base model. In addition, R^2 and the bounding measure were satisfied in this case study. The $MAE_{ensemble}$ (with default settings) was calculated at 56.957 min, and the same metric, when calculated for the same algorithm (without tuning), was at 68.03 min (R^2 between the ensemble and the codal equation was 89%, and it is quite possible to improve the performance of the tuned ensemble by tuning the default settings). The tuned ensemble seems to match a good degree of similarity in terms of trends between PDPs and ALEs with the former codal provision (see Fig. 11). In some features (namely, *k*, *f*, and *L*), a gap is noticed to occur between the PDPs and ALEs obtained from all ML Models and the equation. This gap seems to have a shifting effect, wherein the models capture the trends of the PDPs/ALEs but not the values in some of the involved features.



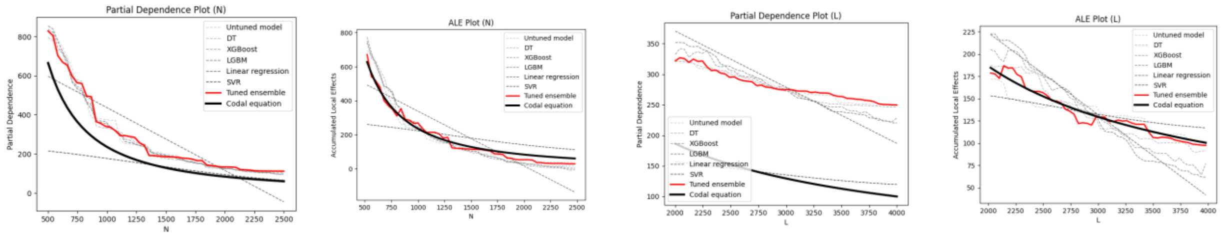


Fig. 11 Comparison for case study no. 3 (in min) [Note: *b* and *d* denote the breadth and depth of timber beams, respectively. The base ML mode: LGBM.]

4.6 Additional case studies based on provisions stemming from a theoretical derivation

These two case studies explore the applicability of the proposed approach to codal provisions that are not empirical in nature but rather derived from first principles.

<u>AISC: Manual Section F6.1. for I-shaped members and channels bent about their minor axis</u>

Section F6.1 in the AISC manual presents the following simple equation to evaluate the nominal flexural strength, M_n, in yielding.

$$M = f_y \times Z \ \leq 1.6\, f_y S_y \tag{13}$$

where, S_y is the elastic section modulus taken about the y-axis.

This simple codal provision has two main features, and the data used in this case study were selected to satisfy the noted condition. XGBoost was selected to be the base ML model with two similarity measures (i.e., Fréchet distance and the Bounding measure). The outcome of this analysis is shown in Fig. 12. Both similarity metrics were satisfied, and the $MAE_{ensemble}$ was calculated at 1637.95 kip.in, and the same metric, when calculated for the same algorithm (without tuning), was at 1659.74 kip.in. Looking at Fig. 12 clearly shows that the tuned ensemble matches the PDP and ALE of the codal equation and outperforms those obtained from the other models.

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

Despite being a simple case study and the need for additional tests to come in future work, this finding seems to indicate the suitability and applicability of the proposed approach to the ability of ML to mimic principle-based codal provisions.

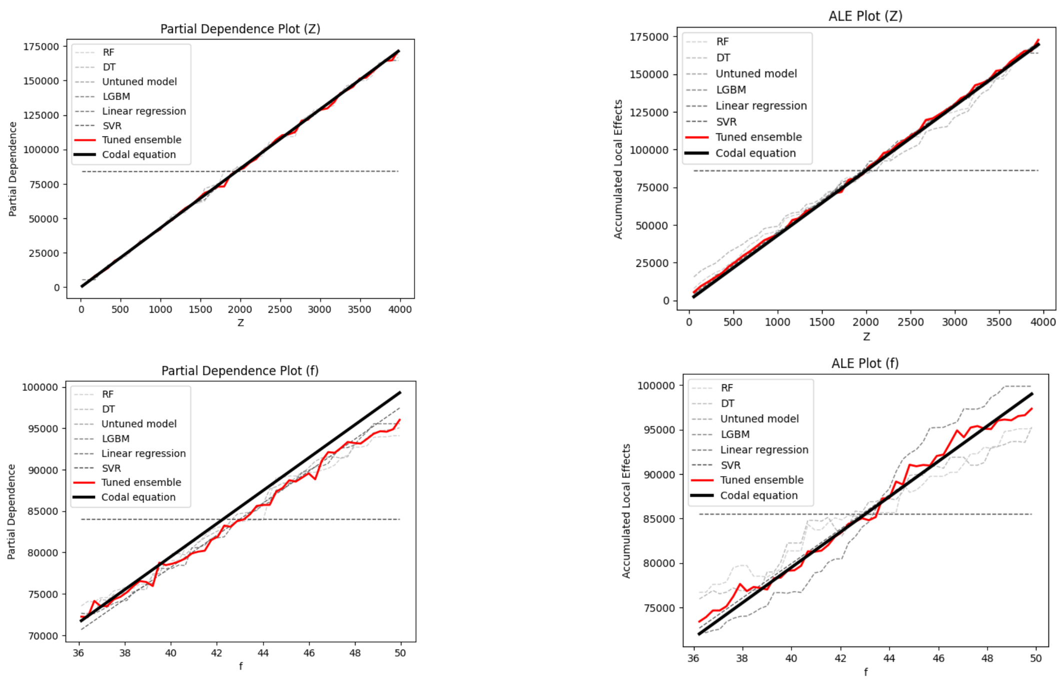


Fig. 12 Comparison for case study no. 6 (in kip.in) [Note: The base ML mode: XGBoost.]

<u>Various building codes: On the elastic deformation of beams</u>

Here, another theoretically derived expression is used as a case study. The equation for elastic deformation in simply supported beams loaded with a point load placed at the midspan is shown below and can be found in a variety of building codes/standards (such as Table 3-23 in the AISC manual):

$$Elastic\ deformation = \frac{PL}{48EI} \tag{14}$$

where, P is the applied load (N), L is the span of the beam (mm), E is the modulus of elasticity (Pa), and I is the moment of inertia (mm^4).

For this case study, the XGBoost model was selected as the base model. All five similarity measures were used, but only one measure (R^2) was satisfied (given default settings). Figure 13 paints a comparison between the generated PDPs and ALEs from the model and the above equation. Despite satisfying one similarity measure, the achieved performance can be deemed acceptable. For completion, the $MAE_{ensemble}$ was calculated at 2.461 mm, and the same metric was 2.391 mm when calculated for the same algorithm (without tuning). R^2 between the ensemble and the equation was 96%.

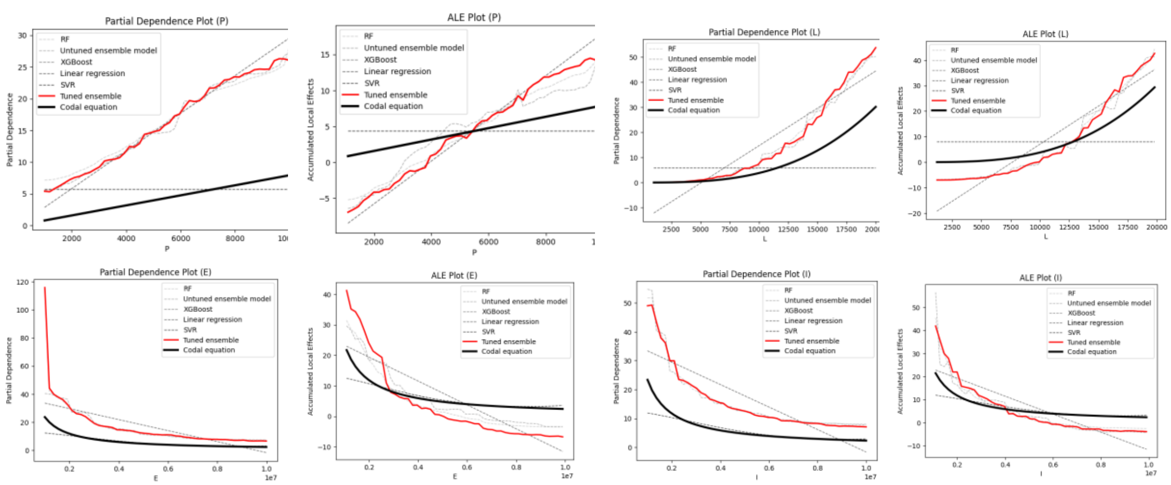


Fig. 13 Comparison for case study no. 7 (in mm) [Note: The base ML mode: XGBoost]

5.0 Additional thoughts, limitations, and future directions

This section documents some of the main limitations of the proposed methodology and shares insights into future research directions.

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

5.1. A note on other matching methods

An engineer might entertain the idea of achieving equivalence through matching feature importance (e.g., a measure of how valuable a feature is in the construction of the model) between the codal provision and the ML model. For example, if a given model's error increase after systematically permuting a feature, then the so-called feature is indeed important since permuting has adversely affected the model's predictions (Altmann et al. 2010).

While this concept could be possible to integrate into the proposed methodology and was indeed successful in the first two studies (yet, these results were not shown in the body of this work), one must keep in mind that feature importance measures the total effect of shuffling a feature on prediction accuracy. While this is a common approach in ML, the perception of feature importance in an equation is somewhat of a nuanced concept simply because all features in a given equation are necessary to hold true (Naser 2023a). Therefore, one could argue that all features in an equation are equally important.

The reader is to note that other explainability measures (i.e., SHAP (SHapley Additive exPlanations (Lundberg and Lee 2017)), LIME (Local Interpretable Model-agnostic Explanations (Ribeiro et al. 2016)), prototypes, etc.) could be explored to build upon the current analysis. The author believes that, despite their computational complexity, such measures could be proven successful and hence can be streamlined and added to the proposed methodology. The author would like to point out that there is room to investigate the influence of model type (i.e., tree-based, ANN, etc.) vs. the degree of explainability. It is possible that some models might fit a phenomenon more easily/accurately/with higher explainability than other models, as such, there is a possibility of a trade-off between accuracy & explainability.

5.2. Limitations of the proposed methodology and future directions

It is important to note that the presented approach can be viewed as complex and somewhat lengthy. As such, this approach may struggle to guarantee the convergence of a ML model that both fits the data well and matches the physical equation's behavior. In reality, the proposed approach converged within 2 minutes of analysis for all the case studies mentioned in this work using a typical desktop that can be found at a common engineering design office. However, in lengthy simulations or multi-phased codal provisions, it is possible that the proposed approach might suffer from instability due to chasing a target that continues to slightly but constantly changes based on the model's own performance.

Fortunately, there are potential solutions to the above problems. One potential approach to attempt to optimize model parameters via gradient-based methods by computing the gradients of the PDPs/ALEs concerning the model parameters. Alternatively, an engineer might consider employing evolutionary or population-based optimization algorithms to search the parameter space for a set that minimizes the discrepancy between the PDPs/ALEs. If neural networks are used, then pre-specified layers can be trained to satisfy similarity metrics separately. Once a layer passes the similarity check, it can then be frozen to maintain it before moving into another measure. This approach could be an alternative to the ensemble approach used herein. It would be interesting to see what other possible solutions (and limitations) this research area might realize in the future.

6.0 Conclusions

Building codes provide definitive standards to guide civil engineering practices and ensure safety. While ML has shown the potential to enhance predictive abilities, its integration into building codes is complicated due to transparency and regulatory approval issues. As civil engineering moves towards a more data-centric future, blending traditional engineering wisdom with innovative ML approaches could unlock new frontiers in designing and constructing safer, more efficient buildings. This work proposes an approach to modernizing building codes via ML. In this approach, empirical and regression-based codal equations can be used to create equivalent ML models that retain the logic and properties of such equations. This approach has been shown to be successful in structural and fire engineering phenomena homed in notable codal provisions.

Data Availability

Some or all data, models, or code that support the findings of this study are available from the corresponding author upon reasonable request.

Conflict of Interest

The author declares no conflict of interest.

Acknowledgment

The authors thank the kind reviewers and editors for their help and support in improving the state of this work.

References

Aladsani, M. A., Burton, H., Abdullah, S. A., and Wallace, J. W. (2022). "Explainable Machine Learning Model for Predicting Drift Capacity of Reinforced Concrete Walls." *ACI Structural Journal*.

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

Altmann, A., Toloşi, L., Sander, O., and Lengauer, T. (2010). "Permutation importance: A corrected feature importance measure." *Bioinformatics*.

Aronov, B., Har-Peled, S., Knauer, C., Wang, Y., and Wenk, C. (2006). "Fréchet distance for curves, revisited." *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

ASCE. (2016). *Minimum Design Loads for Buildings and Other Structures*. *ASCE 7*.

Benjamin, J. R., and Cornell, C. A. (2014). *Probability, Statistics, and Decision for Civil Engineers*.

Bernardin, A., Hoyet, L., Mucherino, A., Gonçalves, D., and Multon, F. (2017). "Normalized euclidean distance matrices for human motion retargeting." *Proceedings - MIG 2017: Motion in Games*.

Breiman, L. (2001). "2001 4_Method_Random_Forest." *Machine Learning*.

Cakiroglu, C., Islam, K., Bekdaş, G., Isikdag, U., and Mangalathu, S. (2022). "Explainable machine learning models for predicting the axial compression capacity of concrete filled steel tubular columns." *Construction and Building Materials*.

Degtyarev, V. V., and Tsavdaridis, K. D. (2022). "Buckling and ultimate load prediction models for perforated steel beams using machine learning algorithms." *Journal of Building Engineering*.

Ellingwood, B. R. (1994). "Probability-based codified design: past accomplishments and future challenges." *Structural Safety*.

Emmert-Streib, F., Yli-Harja, O., and Dehmer, M. (2020). "Explainable artificial intelligence and machine learning: A reality rooted perspective." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*.

Feldman, L. R., and Bartlett, F. M. (2005). "Bond strength variability in pullout specimens with plain reinforcement." *ACI Structural Journal*.

Frank, I., and Todeschini, R. (1994). *The data analysis handbook*.

Franssen, J. M., and Dotreppe, J. C. (2003). "Fire tests and calculation methods for circular concrete columns." *Fire Technology*.

Friedman, J. H. (2001). "Greedy function approximation: A gradient boosting machine." *Annals of Statistics*.

IBC. (2018). *International Building Code*.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T. Y. (2017). "LightGBM: A highly efficient gradient boosting decision tree." *Advances in Neural Information Processing Systems*.

This is a preprint draft. The published article can be found at: https://doi.org/10.1061/JSENDH.STENG-12934.

Please cite this paper as:
Naser M.Z., (2024). Integrating Machine Learning Models to Building Codes and Standards: Establishing Equivalence through Engineering Intuition and Causal Logic. *ASCE Journal of Structural Engineering*. https://doi.org/10.1061/JSENDH.STENG-12934.

Kent, J. T. (1983). "Information gain and a general measure of correlation." *Biometrika*.

Kodur, V. K. R. (1999). "Performance-based fire resistance design of concrete-filled steel columns." *Journal of Constructional Steel Research*, 51(1), 21–36.

Kodur, V., and MacKinnon, D. (2000). "Design of concrete-filled hollow structural steel columns for fire endurance." *Engineering Journal-American Institute of Steel Construction*, 37(1), 13–24.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Priol, R. Le, and Courville, A. (2021). "Out-of-Distribution Generalization via Risk Extrapolation (REx)." PMLR.

Kuchma, D. A., Wei, S., Sanders, D. H., Belarbi, A., and Novak, L. C. (2019). "Development of the one-way shear design provisions of ACI 318-19 for reinforced concrete." *ACI Structural Journal*.

Law, T. (2021). "Beyond Minimum: Proposition for Building Surveyors to Exceed the Minimum Standards of the Construction Code." *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*.

Lie, T. T. (1992). *Structural Fire Protection*. (T. T. Lie, ed.), American Society of Civil Engineers, New York.

Lundberg, S. M., and Lee, S. I. (2017). "A unified approach to interpreting model predictions." *Advances in Neural Information Processing Systems*.

Marasco, S., Fiore, A., Greco, R., Cimellaro, G. P., and Marano, G. C. (2021). "Evolutionary Polynomial Regression Algorithm Enhanced with a Robust Formulation: Application to Shear Strength Prediction of RC Beams without Stirrups." *Journal of Computing in Civil Engineering*.

McNeely, D. J., and Stanley, D. L. (1963). "Tensile Strength of Concrete." *ournal of the American Concrete Institute*, 751–761.

Molnar, C. (2019). "Interpretable Machine Learning. A Guide for Making Black Box Models Explainable." *Book*.

Naser, M. (2023a). *Machine Learning for Civil and Environmental Engineers: A Practical Approach to Data-Driven Analysis, Explainability, and Causality*. Wiley.

Naser, M. Z. (2021a). "Observational Analysis of Fire-Induced Spalling of Concrete through Ensemble Machine Learning and Surrogate Modeling." *Journal of Materials in Civil Engineering*, American Society of Civil Engineers, 33(1), 04020428.

Naser, M. Z. (2021b). "Demystifying Ten Big Ideas and Rules Every Fire Scientist & Engineer Should Know About Blackbox, Whitebox & Causal Artificial Intelligence."

Naser, M. Z. (2023b). *Leveraging artificial intelligence in engineering, management, and safety* *of infrastructure*. (M. Naser, ed.), Taylor & Francis.

Naser, M. Z., Amir, ·, and Alavi, H. (2021). "Error Metrics and Performance Fitness Indicators for Artificial Intelligence and Machine Learning in Engineering and Sciences." *Architecture, Structures and Construction 2021*, Springer, 1, 1–19.

Naser, M. Z., and Ross, B. (2022). "An opinion piece on the dos and don'ts of artificial intelligence in civil engineering and charting a path from data-driven analysis to causal knowledge discovery." *https://doi.org/10.1080/10286608.2022.2049257*, Taylor & Francis.

NDS. (2018). *Technical Report No. 10 Calculating the Fire Resistance of Exposed Wood Members*.

Ollgaard, J. G., Slutter, R. G., and Fisher, J. W. (1971). "Shear Strength of Stud Connectors in Lightweight and Normal-Weight Concrete." *AISC Engineering Journal*.

Paleyes, A., Urma, R. G., and Lawrence, N. D. (2022). "Challenges in Deploying Machine Learning: A Survey of Case Studies." *ACM Computing Surveys,* Association for Computing Machinery, 55(6).

Raphael, J. M. (1984). "Tensile Strength of Concrete." *Journal of the American Concrete Institute*, 81, 158–165.

Reineck, K. H., Kuchma, D. A., Kim, K. S., and Marx, S. (2003). "Shear database for reinforced concrete members without shear reinforcement." *ACI Structural Journal*.

Ribarič, M., Šušteršič, L., and Stefan, J. (2008). "Empirical relationship as a stepping-stone to theory." *Arxiv*.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "'Why should i trust you?' Explaining the predictions of any classifier." *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Riley, R. D., Snell, K. I. E., Ensor, J., Burke, D. L., Harrell, F. E., Moons, K. G. M., and Collins, G. S. (2019). "Minimum sample size for developing a multivariable prediction model: PART II - binary and time-to-event outcomes." *Statistics in Medicine*.

Scikit. (n.d.). "1.4. Support Vector Machines — scikit-learn 0.24.1 documentation." <https://scikit-learn.org/stable/modules/svm.html> (Apr. 5, 2021).

Scikit. (2020). "sklearn.ensemble.GradientBoostingRegressor — scikit-learn 0.24.1 documentation." <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html> (Feb. 9, 2021).

Smeden, M. van, Moons, K. G., Groot, J. A. de, Collins, G. S., Altman, D. G., Eijkemans, M. J., and Reitsma, J. B. (2018). "Sample size for binary logistic prediction models: Beyond events per variable criteria:" *https://doi.org/10.1177/0962280218784726*, SAGE PublicationsSage UK: London, England, 28(8), 2455–2474.

Smith, G. (1986). *Probability and statistics in civil engineering.* Collins, London.

Tapeh, A., and Naser, M. Z. (2022). "Artificial Intelligence, Machine Learning, and Deep Learning in Structural Engineering: A Scientometrics Review of Trends and Best Practices." *Archives of Computational Methods in Engineering*.

Tarawneh, A., Momani, Y., and Alawadi, R. (2021). "Leveraging artificial intelligence for more accurate and reliable predictions of anchors shear breakout capacity in thin concrete members." *Structures*.

Wade, C., Cowles, G., Potter, R., and Sanders, P. (1997). "Concrete Blade Columns in Fire wade - Google Scholar." *Concrete 97 Conference*, Adelaide, Australia.

Windapo, A., and Cattell, K. (2010). "A study of building contractors' compliance with national building regulations in Cape Town." *COBRA 2010 - Construction, Building and Real Estate Research Conference of the Royal Institution of Chartered Surveyors*.

XGBoost Python Package. (2020). "Python Package Introduction — xgboost 1.4.0-SNAPSHOT documentation." <https://xgboost.readthedocs.io/en/latest/python/python_intro.html#early-stopping> (Feb. 10, 2021).

Yahiaoui, A., Dorbani, S., and Yahiaoui, L. (2023). "Machine learning techniques to predict the fundamental period of infilled reinforced concrete frame buildings." *Structures*, Elsevier, 54, 918–927.

Yakubu, S. (2019). "Determinants of non-compliance with structural building code standards in Nigeria." *Proceedings of Institution of Civil Engineers: Management, Procurement and Law*.