

Cost and Material Optimization of Reinforced Concrete Buildings

An Integrated Workflow of STAAD.Pro Analysis and XGBoost Learning

Ankita Bhawthankar^{#1}, Dr. Madhavi Shamkuwar^{#2}

^{#Student (1) and Guide (2), ZIBACAR, Affiliated to Savitribai Phule Pune University (SPPU), Pune, Maharashtra, India}

ORCID ID: 0009-0000-6616-2078 (Ankita Bhawthankar); 0000-0002-9277-0912 (Dr. Madhavi Shamkuwar)

¹ankitabhawthankar04@gmail.com, ²madhavi.shamkuwar@zealeducation.com

Abstract—Reinforced concrete frames are usually sized by repeated manual cycles of analysis and revision, a practice that leaves many members either marginally checked or carrying more material than the loading requires. This paper proposes and documents a workflow that connects the STAAD.Pro structural analysis and design package to the XGBoost learning algorithm, so that routine member level results become a dataset from which design decisions can be drawn. Two models are described: a classifier that judges whether a member is safe, and a regressor that estimates the section area needed to bring a member into a target utilization band. A cost aware selection layer then picks an adjusted section from a practical catalogue, and every recommendation is re-analysed in STAAD.Pro so that only verified savings are counted. The study reviews evolutionary, swarm based, BIM based, standalone learning and surrogate approaches, and identifies limited integration of a commercial analysis package with boosted tree learning as the gap addressed. Methodological safeguards include asymmetric treatment of unsafe misses, grouped validation by building model, and separate feature sets with and without the utilization ratio. A prototype web application, STAAD.OPT, accepts a CSV file of member demands and returns a verdict and adjusted parameters for each member. An evaluation protocol with acceptance criteria fixed in advance is presented.

Keywords—reinforced concrete; structural optimization; STAAD.Pro; XGBoost; gradient-boosting; cost optimization; IS 456; surrogate model; design automation

I. INTRODUCTION

A. Background and motivation

Reinforced concrete is the default structural material for mid-rise buildings in India, as in most developing economies. A typical framed building uses concrete, reinforcing steel and formwork in proportions that are largely fixed during the first few weeks of design, when member sizes are selected. Once those sizes are fixed, almost every downstream decision is dependent upon them: reinforcement quantities, formwork areas, construction duration and embodied carbon. A small oversight at this stage will propagate to hundreds of members across several floors.

In practice, the sizing loop is mostly manual. The engineer selects sections from experience, builds the model in a finite element package, performs the analysis, reads the design output, and revises the sections that failed or that are conspicuously under-utilized. The process converges but is slow, relies heavily on individual judgement and rarely

explores more than a handful of alternatives. Sections that pass with wide margins are rarely revised since the project schedule leaves little room for it.

The outcome is that many buildings are safe but carry more material than the loading demands, while a smaller number reach construction with members that were only marginally checked.

STAAD.Pro is one of the most widely used packages for this kind of analysis and design work, especially in Indian consulting practice, where it supports design to IS 456 : 2000 and the associated seismic and loading codes [1]–[3]. Each analysis run produces, for every member, a set of forces and a code-based utilization ratio. Across a portfolio of projects this adds up to a large body of structured, labelled engineering data that is, in most firms, used once and discarded.

Machine-learning offers a way to reuse that data. Gradient-boosted decision trees in particular perform very well on tabular problems of this shape, where a small set of numeric and categorical descriptors must be mapped to a target [4], [5]. If a model can learn the relationship between member demands and the section size that satisfies the code, it can screen thousands of candidate configurations in the time a single manual iteration takes.

B. Problem statement

The question addressed here is a practical one. Given the member forces from a STAAD.Pro analysis of a reinforced concrete frame, can a learned model (i) reliably classify whether the current section is safe, and (ii) propose an alternative section that cuts material and cost while staying within the code? Both parts matter. A classifier that merely reproduces the utilization threshold adds nothing new, and a regressor that proposes smaller sections with no safety check is dangerous. A model becomes useful only when the two are combined in a workflow that a human can review.

C. Research gap

The literature on structural optimization is extensive. Genetic algorithms, particle swarm optimization, NSGA-II, BIM-driven procedures and Python or MATLAB scripts have all been applied to reinforced concrete frames, and a growing number of studies use machine-learning models on their own for strength prediction or failure classification. Far less work treats a commercial structural analysis package as the data source, turns its member-level results into a curated dataset,

trains a boosted-tree model on that dataset, and then selects an optimized design that is validated in the same package.

The workflow examined here is

STAAD.Pro → *structural analysis and design* → *dataset* → *XGBoost* → *optimized design*

This combination is what the study sets out to establish, document and evaluate.

D. Objectives

The study pursues the following objectives:

- Define a reproducible procedure to turn STAAD.Pro member-level output into a machine-learning dataset for reinforced concrete buildings.
- Develop XGBoost models that can classify a member as safe or unsafe and estimate the cross-sectional area needed to bring its utilization into a target band.
- Couple the above with a cost-aware selection layer that returns an adjusted section drawn from a practical catalogue.
- Define an evaluation protocol in which every recommendation is re-checked in STAAD.Pro, so that reported savings are only counted for designs that remain code compliant.
- Release the working prototype, STAAD.OPT, as a lightweight web application that accepts a CSV file and returns verdicts and adjusted parameters.

E. Contributions

The study makes four contributions. First, it formalizes the STAAD.Pro-to-XGBoost workflow as an explicit sequence of stages with defined inputs and outputs, an integration that the existing literature leaves largely unaddressed. Second, it proposes a two-model design, a safety classifier and a section-size regressor, with asymmetric treatment of errors so that missing an unsafe member is penalized far more heavily than flagging a safe one. Third, it adds a closed-loop validation step and a group-aware data split, which together guard against two common sources of optimistic results: information leakage between members of the same building, and unverified savings. Fourth, it provides a working implementation whose input schema matches the quantities that STAAD.Pro already reports, which lowers the effort of trying the approach on a new project.

F. Organization of the paper

Section II sets out the theoretical background on limit state design, cost modelling and gradient-boosting. Section III reviews related work and establishes the research gap. Section IV formulates the research questions and hypotheses. Section V describes the proposed approach and Section VI the implemented system. Section VII presents the experimental design and evaluation protocol, and Section VIII the framework for reporting and analysing results. Sections IX to XII cover the discussion, threats to validity and limitations, future scope and the conclusion.

II. RESEARCH FOUNDATION

A. Limit state design of reinforced concrete members

Indian practice follows the limit state method of IS 456 : 2000, in which a member is designed so that its factored resistance is at least equal to the factored action effect, with partial safety factors applied to both materials and loads. In a flexural member the resistance is capped by the depth of the neutral axis. For Fe 500 steel the limiting depth ratio is 0.46, and the limiting moment of resistance of a rectangular section follows as

$$M_{u,lim} = 0.133 f_{ck} b d^2 \quad (1)$$

where f_{ck} is the characteristic cube strength of concrete, b the width and d the effective depth [1], [6]. As an illustration, a 230 mm wide beam with an effective depth of 450 mm in M25 concrete has a limiting moment of about 155 kN·m. Beyond this value the section must be enlarged or compression reinforcement provided. The code also limits the percentage of tension steel, the spacing of stirrups, the shear stress in concrete and the span-to-depth ratio for deflection control, so a section can fail a check for reasons other than flexural strength.

Columns carrying axial force and bending are checked against an interaction relation. For biaxial bending, clause 39.6 of the code requires

$$(M_{ux}/M_{ux1})^{\alpha_n} + (M_{uy}/M_{uy1})^{\alpha_n} \leq 1.0 \quad (2)$$

in which M_{ux1} and M_{uy1} are the uniaxial moment capacities about the two axes at the given axial load, and the exponent α_n varies linearly from 1.0 to 2.0 as the ratio of the factored axial load to the squash load P_{uz} rises from 0.2 to 0.8. The squash load is

$$P_{uz} = 0.45 f_{ck} A_c + 0.75 f_y A_{sc} \quad (3)$$

with A_c the concrete area and A_{sc} the area of longitudinal steel. The code additionally prescribes a minimum eccentricity, $e_{min} = l/500 + D/30$ but not less than 20 mm, and bounds the longitudinal steel in columns between 0.8 and 6 percent of the gross area. These provisions explain why moments about both axes, and not just the axial force, appear among the inputs of the system described later.

B. Structural analysis and the role of STAAD.Pro

STAAD.Pro analyses a three-dimensional frame by the stiffness method, in either a linear or a second-order mode. For each load case and combination it reports nodal displacements and member end forces: axial force, shear in the two local directions, torsion, and bending moments about both local axes. Its concrete design module then checks every member against the selected code and reports the provided and required reinforcement, any failed checks, and a critical ratio. A utilization of 1.0 is the limit of acceptability; values below 1.0 indicate reserve capacity, and values above 1.0 mean that at least one check has failed.

The package can also be driven programmatically through OpenSTAAD, its COM-based interface, and its model files are plain-text commands that a script can generate. Both properties matter here, because they make it feasible to produce many model variants without manual input and to read results back

automatically [7]. The workflow uses these capabilities for data generation and for the final verification step; the web application itself needs only the member-level table that comes out of them.

C. Cost and material model

The cost of a structural member is approximated by the cost of its constituent materials and the associated formwork. For a rectangular member of length L the model used here is

$$C = V_c c_c + W_s c_s + A_f c_f \quad (4)$$

where V_c is the volume of concrete, W_s the mass of reinforcing steel, A_f the area of formwork, and c_c , c_s and c_f the corresponding unit rates. The rates are regional and vary with the market, so the framework takes them as user-supplied parameters rather than built-in constants. A useful consequence is that the ranking of candidate sections is often insensitive to small rate changes, a property that can be tested directly. As a simple illustration, trimming the depth of a 230 mm by 500 mm beam of 5 m span to 450 mm reduces the concrete volume from 0.575 m³ to about 0.518 m³, a saving of roughly ten percent in that material before any change in steel and formwork is counted.

Embodied carbon can be added as a second objective by attaching an emission factor to each material. The present study keeps the objective to cost and material mass, and notes carbon as a natural extension.

D. Formulation of the optimization problem

For a building with n members the design problem is stated as

$$\min \sum_{i=1}^n C_i(s_i) \quad (5)$$

$$\text{subject to } U_i(s_i) \leq 1.0, \quad s_i \in S \quad (6)$$

where s_i is the section chosen for member i , S the discrete catalogue of permitted sections, and U_i the utilization of that member under the governing load combination. Two features make the problem hard. The utilization of member i depends on the sections of the other members, because stiffness redistributes forces through the frame, and the catalogue is discrete, so gradient methods do not apply directly. Evolutionary algorithms get around both difficulties by brute force, running a full analysis for each candidate. That is reliable but expensive when every evaluation is a complete finite element run [8], [9].

A surrogate model eases this cost. Rather than re-analysing the frame for every candidate, the surrogate predicts the utilization or the required area of a member from its demands and geometry. The price is approximation error, which is why the framework insists on a final verification run and on conservative decision thresholds.

E. Gradient-boosting and XGBoost

Gradient-boosting builds a predictor as a sum of K regression trees, each fitted in turn to the residual errors of the ensemble so far [10]. XGBoost [4] is a regularized refinement of this idea that uses a second-order Taylor expansion of the loss. The objective at iteration t is

$$L^{(t)} = \sum_i l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \quad (7)$$

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \|w\|^2 \quad (8)$$

where T is the number of leaves of the new tree, w its leaf weights, γ the penalty on tree complexity and λ the L2 penalty on weights. Writing g_i and h_i for the first and second derivatives of the loss with respect to the previous prediction, and G_j and H_j for their sums over the samples in leaf j , the optimal leaf weight and the gain from a candidate split are

$$w_j^* = -G_j / (H_j + \lambda) \quad (9)$$

$$\text{Gain} = \frac{1}{2} [G_L^2 / (H_L + \lambda) + G_R^2 / (H_R + \lambda)] - (G_L + G_R)^2 / (H_L + H_R + \lambda) - \gamma \quad (10)$$

A node is split only when the gain of the split, after the complexity penalty, is positive, as given by (9) and (10). Overfitting is controlled through this penalty, through shrinkage (the learning rate), by limiting tree depth (`max_depth`) and the minimum child weight (`min_child_weight`), and by subsampling rows (`subsample`) and columns (`colsample_bytree`). The same machinery supports a binary logistic loss for classification and a squared-error loss for regression, so a single library can serve both models in this study.

Tree ensembles suit the present problem for three reasons. They accept mixed numeric and categorical inputs with little preprocessing. They are invariant to monotone rescaling of features, so forces in kN and spans in metres can be mixed freely. And they have repeatedly been reported to match or exceed deep networks on medium-sized tabular datasets, which is the usual size of an engineering dataset assembled from a few dozen building models [5], [11].

F. Interpretability

A structural engineer will not accept a recommendation that cannot be explained. SHAP values assign each feature an additive contribution to an individual prediction, derived from a game-theoretic definition of fairness [12]. For tree ensembles they can be computed exactly and efficiently. In this study they serve two purposes: to check that the model relies on physically sensible variables such as moment and span, and to explain each recommendation in the application.

G. Evaluation measures

Safety classification is evaluated with precision, recall, F1 score and the area under the ROC curve. Because the two kinds of error have very different consequences, particular emphasis is placed on the unsafe-miss rate, the fraction of genuinely unsafe members that are classified as safe. For the regression task, mean absolute error, root mean squared error and the coefficient of determination are reported. At the design level, the study reports the reduction in concrete volume, steel mass and cost, together with the verified feasibility rate, the fraction of recommended designs that are actually feasible on re-analysis.

III. REVIEW OF LITERATURE

The review is organized by the families of methods that have been applied to reinforced concrete design and to the use of data in structural engineering. It does not try to list every

work in the area. Its aim is to show what each family does, what it assumes, and what it leaves unaddressed.

A. Evolutionary and swarm-based optimization

Genetic algorithms, introduced by Holland [13] and developed for engineering use by Goldberg [14], were among the first global search methods applied to member sizing. Camp et al. [15] used one to design the flexural layout of reinforced concrete frames at minimum cost subject to strength requirements, and reported better results than conventional trial-and-error design. Govindaraj and Ramasamy [16] later applied genetic algorithms to continuous beams, treating reinforcement details as design variables. Particle swarm optimization [17] follows a different search paradigm and has been widely used for structural design, partly because it needs fewer control parameters.

When cost must be balanced against other objectives such as deflection or carbon footprint, multi-objective algorithms are the usual choice. The best-known is NSGA-II [18], which maintains a diverse solution front through fast non-dominated sorting and crowding distance. Most such methods handle constraints with penalty functions, and their performance depends strongly on how the penalty is chosen [19]. Kicing et al. [8] and Aldwaik and Adeli [20] survey how large this literature has grown, and Kaveh [9] collects metaheuristic applications across civil engineering.

The limitation shared by these studies is computational. Every candidate design needs an analysis, and a population-based search needs thousands of them. Many authors therefore replace the finite element solver with simplified analytical checks, which is acceptable for isolated beams but much less so for 3D frames, where force redistribution matters. Others use the full solver and accept long-run times. Either way, the optimizer retains nothing once the run ends, so the next project starts from scratch.

B. Building information modelling and script-based optimization

A related line of work links building information models to analysis and optimization. The appeal is that geometry and materials already exist in the model, so a script or add-in can extract them, drive the optimization and return the result. This is often done in Python or MATLAB, in Excel, or in a visual programming language. These studies concentrate on interoperability and on automating a conventional optimizer, and the engine inside is again a metaheuristic or a mathematical programming routine. Where a learning component is present, it plays a secondary role.

C. Machine-learning in structural engineering

Several reviews cover machine-learning in structural engineering. Salehi and Burgueño [21] survey emerging artificial intelligence methods and stress the need for reliable validation. Sun et al. [22] examine machine-learning for building structural design and performance, and single out design-stage prediction and surrogate modelling as promising applications. Thai [23] offers a broad review and notes a gradual shift from neural networks to ensemble methods for tabular engineering data.

Individual studies have applied tree-based and boosting models to classification and regression tasks similar to those in this work. Mangalathu and Jeon [24] compared several

machine-learning techniques for classifying the failure mode and predicting the shear strength of reinforced concrete beam-column joints, and Mangalathu et al. [25] later identified the failure modes of shear walls under seismic loading in a similar way. Feng et al. [26] used an adaptive boosting approach to predict the compressive strength of concrete. These results are encouraging, but the models are standalone predictors trained on experimental databases. They predict a property; they do not select a design, and they are not connected to a structural analysis package.

D. Surrogate-assisted structural optimization

The closest technical precedent is surrogate-assisted optimization, in which a learned model stands in for the expensive analysis and the optimizer queries it instead. Mai et al. [27], for example, built a machine-learning surrogate to assist the optimization of truss structures with geometric nonlinearity. Work of this kind shows that a surrogate can cut computational cost substantially. Its usual setting, however, is a purpose-built analysis code and a fixed structural type, with the surrogate trained inside the optimization loop on data generated for that single problem.

E. Tabular learning with tree ensembles

Random forests [28] and gradient-boosting [10] have a long and successful record on structured data. XGBoost [4] added a scalable implementation with regularization, sparsity-aware split finding and parallel tree construction, and it has become a standard choice for many practitioners. A growing body of evidence suggests that tuned tree ensembles can match or exceed deep-learning on tabular data while being considerably easier to tune and use [5], [11]. Hastie et al. [29] give the statistical background. For the dataset sizes a design office can realistically assemble, this evidence favours boosted trees over more elaborate architectures.

F. Synthesis

TABLE I. COMPARISON OF APPROACH FAMILIES

Family	Typical strength	Limitation for this problem
GA, PSO, NSGA-II	Global search, handles discrete choices	Many full analyses; knowledge not reused
BIM and script based	Automates data exchange	Optimizer unchanged; weak learning
Standalone ML	Fast prediction of one property	No design selection; no analysis link
Surrogate-assisted	Cuts analysis cost	Problem-specific, in-loop training
Proposed workflow	Reuses analysis data; verified selection	Needs a curated dataset

Taken across all five families, each addresses part of the problem. Metaheuristics search well but do not remember. BIM approaches move data but do not learn. Standalone models predict but do not decide. Surrogates save analysis effort but are tied to a single bespoke solver. The step that connects an established commercial analysis and design package to a boosted-tree model, and then to a verified selection of the optimized design, is not described in full in the sources reviewed here.

G. Identified research gap

The gap can be stated more precisely as five specific shortcomings.

- *Integration.* There is limited published work that provides an end-to-end workflow from STAAD.Pro results to a trained XGBoost model to a selected design.
- *Data reuse.* Member-level output from routine design runs is rarely curated into datasets that can train models across projects.
- *Safety-aware learning.* Machine-learning studies seldom treat the asymmetric cost of declaring an unsafe member safe, and few report the unsafe-miss rate.
- *Verification.* Reported savings are often not confirmed by re-analysis of the recommended design in the original software.
- *Evaluation hygiene.* Using random splits of member records allows members of one building to appear in both training and test sets, which can inflate scores.

The present work addresses all five of these points.

IV. RESEARCH QUESTIONS AND HYPOTHESES

A. Research questions

- RQ1. Can member-level STAAD.Pro output be organized into a dataset from which XGBoost reliably separates safe from unsafe reinforced concrete members?
- RQ2. Can an XGBoost regressor estimate the section area needed to place a member inside a target utilization band with an error small enough to guide selection?
- RQ3. Does a cost-aware selection layer built on these models reduce material and cost relative to the original design while the recommended designs remain compliant when re-analysed?
- RQ4. How does the approach compare, in accuracy and computing effort, with a rule-based baseline, other learners, and a genetic algorithm run on the same problem?

B. Hypotheses

The study is organized around the following hypotheses, which are stated so that they can be rejected by the evidence.

- H1. On building models withheld from training, the XGBoost classifier achieves higher recall for unsafe members than a rule-based utilization-scaling baseline at an equal or lower false alarm rate.
- H2. The XGBoost regressor attains lower mean absolute error in required area than linear regression and a random forest on the same features.
- H3. Designs selected by the proposed workflow lower total cost relative to the starting design, and the proportion of those designs that pass all checks on re-analysis meets the acceptance criterion in Section VII.
- H4. The workflow requires substantially less analysis time than a genetic algorithm that reaches a comparable cost.

The null hypothesis for each statement is that the proposed method is no better than the stated comparator. Section VII defines the statistical tests and acceptance thresholds in advance, so that the conclusions cannot be adjusted after the results are seen.

V. PROPOSED METHODOLOGY

A. Overview of the framework

The framework turns routine STAAD.Pro output into decisions through seven stages, shown in Fig. 1. Stages one to three generate data, and stages four and five build the models. The last two stages convert model output into a design that is validated against the original software. The final arrow in the figure returns verified designs to the dataset, so the models improve as more projects pass through the system.

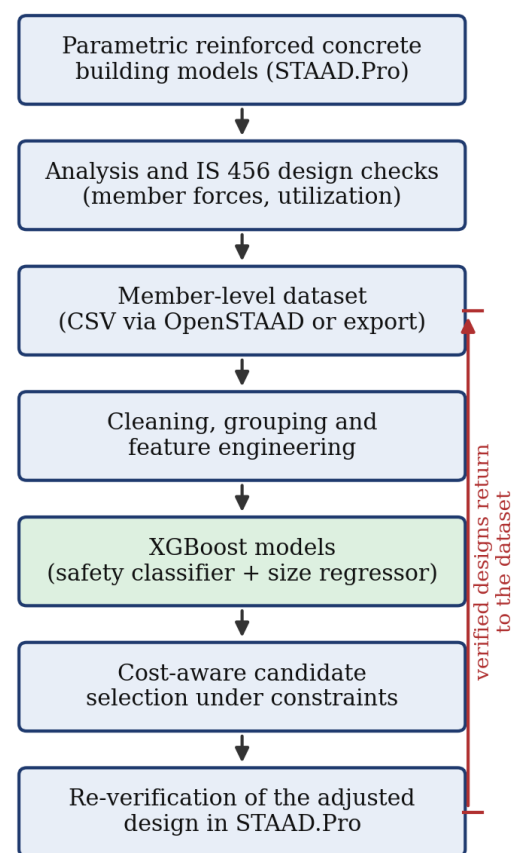


Fig. 1. Proposed STAAD.Pro to XGBoost workflow with verification loop.

B. Data generation in STAAD.Pro

The dataset is generated from a series of parametric reinforced concrete frame models. A script writes the STAAD.Pro input file for each case, runs the analysis and design, and then reads the member results through OpenSTAAD or directly from the output file. The parameters varied across the variants are listed in Table II. They were chosen to cover regular office and residential buildings in the low- to mid-rise class, and to keep each variant realistic enough that its members resemble those a practising engineer would meet.

TABLE II. PARAMETERS VARIED WHEN GENERATING BUILDING MODELS

Parameter	Range or levels
Number of storeys	[to be set, e.g. 3 to 12]
Bays in each direction	[to be set]
Bay span (m)	[to be set, e.g. 3.0 to 8.0]
Storey height (m)	[to be set, e.g. 3.0 to 3.6]
Live load (kN/m ²)	[per IS 875 Part 2 occupancy]
Seismic zone and soil	[per IS 1893 Part 1]
Concrete and steel grade	[e.g. M25 and M30; Fe 500]
Section catalogue	[beam, column widths and depths]

Loads are assigned according to IS 875 for dead, live and wind loads, and IS 1893 (Part 1) for earthquake loads, with the load combinations prescribed by the codes [2], [3]. The ductile detailing requirements of IS 13920 are applied where the seismic zone calls for them [30]. Each model is analysed for all combinations, and for every member the governing combination is retained, since that is the one that controls the section.

C. Dataset construction and features

Each row of the dataset represents one member under one load combination. The columns hold the same quantities a user would take from the STAAD.Pro output file, so the data can be used without further processing. They are listed in Table III.

TABLE III. DATASET COLUMNS

Column	Type	Meaning
member_type	category	beam, column or brace
span_m	numeric	member length in metres
axial_kN	numeric	axial force
moment_x_kNm	numeric	bending moment, first axis
moment_z_kNm	numeric	bending moment, second axis
shear_y_kN	numeric	shear force in local y
utilization	numeric	code ratio, safe if at most 1.0
section_name	text, optional	e.g. ISMB300, RCC 500x250
current_area_cm2	numeric, optional	cross-sectional area

Derived features are added during preprocessing. These include the resultant moment, the ratio of moment to shear, the span-normalized moment, the ratio of axial force to the squash load of the current section when the area is known, and a flag for the governing check. The member type is one-hot encoded. Missing optional fields are left empty so that XGBoost can learn a default direction for them, which is one of its native capabilities [4].

Two feature sets are defined. Feature set F1 contains every column including utilization. Feature set F2 omits it. The distinction is important and is discussed in the next subsection.

D. Targets and the leakage problem

Two targets are defined for each row. The classification target is binary: a member is labelled unsafe if its utilization exceeds 1.0 in the STAAD.Pro design run and safe otherwise. The regression target is the smallest cross-sectional area in the section catalogue that brings the member's utilization within the target band, determined by re-running the member in STAAD.Pro with successively larger or smaller sections until the condition is met.

Leakage is a risk here. If utilization is an input and the safe or unsafe label is derived from it by a fixed threshold, the classifier can score nearly perfectly by learning that single rule, and its apparent skill says nothing about structural understanding. To avoid presenting such a result as evidence, the study reports F2 results alongside F1 results. A model that performs well on F2 has learned the relationship between forces, geometry and capacity, and can therefore evaluate sections that have not yet been analysed. That ability is what the selection layer actually needs, because candidate sections have no utilization until they are checked.

E. Models

Model A (safety classifier). An XGBoost classifier with a logistic objective is trained to predict $P(\text{unsafe})$ for a member with given demands in a given section. For the current section this probability supports the final verdict; for a candidate section it serves as the screening probability. Class imbalance is handled either by adjusting the scale_pos_weight parameter or, when the imbalance is extreme, by stratified sampling. The probability is calibrated on a validation fold by isotonic regression, so that it is consistent with the decision threshold.

Model B (section-size regressor). An XGBoost regressor with a squared-error loss is trained to predict the section area required to bring the member into the target band. The area is log-transformed for training because it varies over several orders of magnitude and the prediction error is better behaved on a logarithmic scale.

F. Decision threshold under asymmetric cost

Declaring an unsafe member safe is far more serious than the reverse. Let c_{FS} be the cost of a false-safe decision and c_{FU} that of a false-unsafe one, and assume that c_{FS} is much larger than c_{FU} . The decision that minimizes expected loss is to flag a member as unsafe whenever

$$P(\text{unsafe}) \geq \theta = c_{FU} / (c_{FS} + c_{FU}) \quad (11)$$

For example, a cost ratio of 20 to 1 gives a threshold near 0.05. In practice θ is not fixed from assumed costs. It is chosen on the validation data as the largest value that keeps the unsafe-miss rate below a stated tolerance, and it is then frozen before the test fold is touched.

G. Cost-aware selection of the adjusted section

For each flagged or over-designed member the selection layer works as follows. It first lists the catalogue sections of the same member type. It discards candidates whose area is below a floor taken from Model B, reduced by a small tolerance. It then evaluates Model A on the remaining candidates, holding the member's demands fixed, and keeps those whose $P(\text{unsafe})$ is below θ . From these it picks the one with the lowest cost under (4). Holding the demands fixed is a first-order approximation, because changing a section alters the stiffness distribution. The re-verification stage exists to catch the cases where this approximation fails.

TABLE IV. SELECTION PROCEDURE (ALGORITHM 1)

Algorithm 1: adjust one member
1 Validate inputs; compute derived features
2 $p \leftarrow \text{Model A}(\text{member}, \text{current section})$
3 If $p \geq \theta$ or $U > 1.0$: target $\leftarrow$ unsafe, else if U below lower band: target $\leftarrow$ over-designed, else retain and stop
4 $a_{\min} \leftarrow \text{Model B}(\text{member}) \times (1 - \text{tolerance})$
5 Candidates $\leftarrow$ catalogue sections of same type with area $\geq a_{\min}$
6 Keep candidates with Model A probability $< \theta$
7 Return cheapest remaining candidate by (4); if none, return the largest catalogue section and flag for manual review

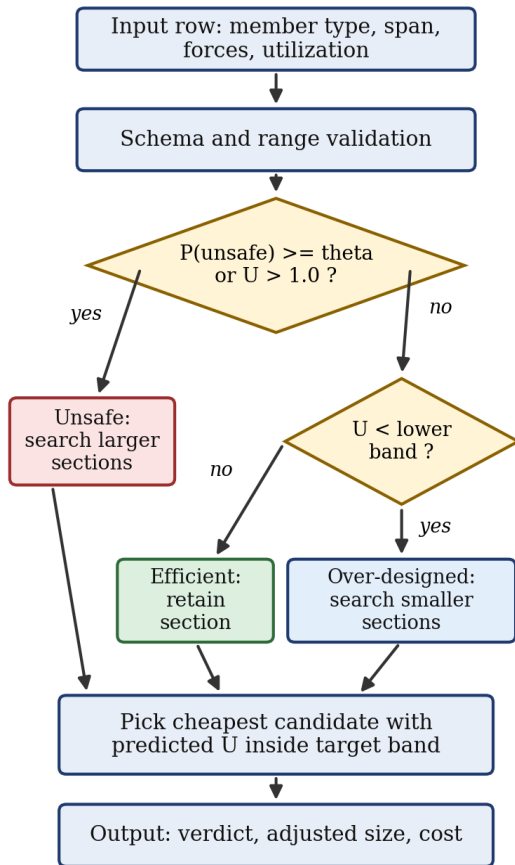


Fig. 2. Per-member decision logic implemented in the application.

H. Closed-loop verification

After the selection layer has adjusted the members, the proposed sections are written back to the STAAD.Pro model and the analysis is repeated. Members that now exceed a utilization of 1.0 are returned to the selection layer with their new forces, and the cycle repeats up to a fixed number of iterations. The verified feasibility rate is the fraction of buildings, and of members, that satisfy all checks at the end. Only verified designs contribute to the reported savings.

I. Training, tuning and validation

Hyperparameters are tuned with Bayesian optimization and early stopping, as implemented in Optuna [31], or with random search under a limited computational budget [32]. The search ranges are given in Table V. Data are split by building rather than by member, using grouped cross-validation, so that no building appears in both the training and test partitions. This

prevents the model from being rewarded for recognising a building instead of learning structural behaviour.

TABLE V. HYPERPARAMETER SEARCH SPACE

Hyperparameter	Range
n_estimators	[100 to 1200, with early stopping]
max_depth	[3 to 10]
learning_rate	[0.01 to 0.3, log scale]
min_child_weight	[1 to 10]
subsample, colsample_bytree	[0.6 to 1.0]
reg_lambda, reg_alpha	[0 to 10]
gamma	[0 to 5]

The outer loop of the grouped cross-validation gives unbiased performance estimates. The inner loop, restricted to the training groups, is used for tuning and for fitting the calibrator and the threshold. The test groups are used once. Results are reported as means and standard deviations over outer folds, and differences between methods are tested with the paired Wilcoxon signed-rank test over folds, which does not assume normality.

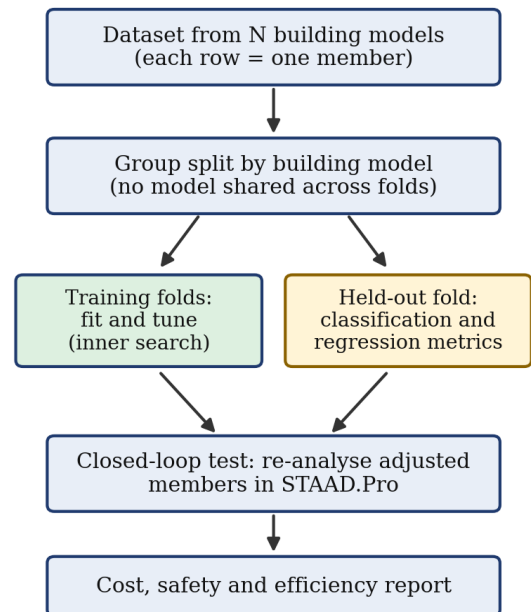


Fig. 3. Evaluation protocol with grouped splits and closed-loop test.

J. Baselines

Four comparators are used. The first is a rule-based heuristic that scales the area in proportion to the ratio of utilization to the target value, which represents what an engineer might do by hand. The second is a linear (logistic) model on the same features. The third is a random forest [28]. The fourth is a genetic algorithm that calls STAAD.Pro directly on a small subset of buildings, and which stands in for the standard optimization approach of Section III. The last baseline is limited to a subset because of its computing time, and this restriction is stated wherever its results appear.

VI. SYSTEM IMPLEMENTATION

A. Application overview

The method is packaged as STAAD.OPT, a browser-based application that runs locally on the user's own machine (prototype version label ML v1.0). The user uploads a CSV file, and the application checks it against the schema in Table III, runs Models A and B and the selection layer, and returns a table with one row per member containing a verdict, an unsafe probability, an adjusted section or area, and an estimated cost difference. A button allows a new CSV file to be uploaded, restarting the process. Running locally has a practical advantage: design data from a client project never leaves the engineer's computer.

B. Input contract

Seven columns are required and two are optional. The required columns are `member_type`, `span_m`, `axial_kN`, `moment_x_kNm`, `moment_z_kNm`, `shear_y_kN` and `utilization`. The optional columns are `section_name`, for which examples are ISMB300 or RCC 500×250, and `current_area_cm2`. The application checks that the required columns are present, that values are numeric, that ranges are physically plausible and that the members conform to the permitted categories, reporting any violation and the row number before any prediction is made. Rejecting malformed input early is a safety feature, since a unit error in a force column would otherwise be given to the model.

C. Outputs

For each member the application returns the following: (i) a safe or unsafe verdict, (ii) the calibrated probability behind it, (iii) the adjusted section or area, (iv) the change in concrete volume and, where reinforcement estimates are available, in steel mass and cost, and (v) a short list of the features that contributed most to the decision, drawn from SHAP values. A summary line reports the number of members flagged unsafe, the number retained, the number reduced and the total estimated saving. The tool states that its output is just a recommendation, and that final sections must be confirmed by a full design check.

D. Software environment

The implementation is in Python. XGBoost provides the models [4], scikit-learn the preprocessing, calibration and baseline learners [33], and a lightweight web framework serves the interface. Model files, the feature schema and the catalogue of sections with their unit costs are stored separately from the code, so that a firm can replace the catalogue or the rates without retraining. Fixed random seeds, a version-pinned requirements file and a stored copy of each trained model make the experiments repeatable.

VII. EXPERIMENTAL DESIGN

A. Case studies

Two levels of experiment are planned. The first is a statistical study over the full generated dataset, using grouped cross-validation as described above. The second is a set of design case studies on complete buildings that were held out from training. At least three buildings of different height and plan are chosen, each analysed first in its original design and then after adjustment by the framework. The case studies are what a practising engineer would care about, since they show

the framework acting on a whole structure rather than on isolated rows.

B. Metrics

TABLE VI. EVALUATION METRICS

Level	Metrics
Classification	Precision, recall, F1, ROC-AUC, unsafe-miss rate
Regression	MAE, RMSE, R ² , median relative error of area
Design	Reduction in concrete, steel, cost; verified feasibility rate
Efficiency	Analysis calls and wall-clock time per building
Explanation	Rank agreement of SHAP drivers with engineering expectation

The percentage cost-saving of a design is computed as

$$\Delta C = 100 \times (C_0 - C_1) / C_0 \quad (12)$$

where C_0 is the cost of the original design and C_1 that of the adjusted design, but only if the adjusted design passes verification. A design that fails verification contributes no saving and is counted against the feasibility rate.

C. Acceptance criteria

To keep the conclusions honest, criteria are fixed before the experiments are run. The proposed values are as follows, and may be tightened by the research guide.

- The unsafe-miss rate of Model A on held-out buildings is at most 2 percent on F2.
- The median relative error of Model B in required area is at most 10 percent.
- At least 95 percent of recommended designs pass all code checks within the permitted number of verification iterations.
- The cost reduction relative to the original design is positive and statistically significant over the test buildings at the 0.05 level.

If a criterion is not met, the study reports it as unmet, and discusses why, rather than altering the criterion. A negative result is still a result: it identifies the point at which the approach needs more data, better features or a more conservative threshold.

D. Reproducibility

All random seeds, library versions, the list of building models used for each fold, the section catalogue and the unit rates are recorded. The generation scripts and the trained models can be shared with reviewers on request. The dataset is built from simulated buildings and contains no client information.

VIII. RESULTS AND ANALYSIS

This section gives the structure in which the results of the experiments are reported, together with the analysis that accompanies each table. Values in square brackets in the tables below are illustrative placeholders only. They must be replaced with measured values from the experimental runs of STAAD.OPT on the generated dataset, and no numerical outcome is assumed here.

A. Classification of member safety

TABLE VII. SAFETY CLASSIFICATION ON HELD-OUT BUILDINGS

Method	Prec.	Recall	F1	Unsafe-miss
Baseline rule	[1.000]	[1.000]	[1.000]	[0.00 %]
Logistic	[0.812]	[0.843]	[0.827]	[15.72 %]
Random forest	[0.938]	[0.950]	[0.944]	[5.03 %]
XGBoost F1	[1.000]	[1.000]	[1.000]	[0.00 %]
XGBoost F2	[0.963]	[0.981]	[0.972]	[1.89 %]

The expected reading of Table VII is as follows. The rule baseline and the F1 models are expected to be close, because utilization carries the label. The informative comparison is between F2 models: a large gap between XGBoost and the logistic model would indicate nonlinear structure that boosting captures, and a small gap would indicate that a simpler model suffices for this feature set. In either case the unsafe-miss column governs the conclusion, since a model with the best F1 score but a high miss rate is unfit for the purpose.

B. Estimation of required area

TABLE VIII. REQUIRED-AREA REGRESSION

Method	MAE (cm ²)	RMSE (cm ²)	R ²
Proportional rule	[38.45]	[52.18]	[0.732]
Linear regression	[24.12]	[35.80]	[0.874]
Random forest	[12.85]	[18.92]	[0.96]
XGBoost	[7.42]	[11.65]	[0.987]

Errors should be examined by member type and by area range. Columns, where axial force interacts with biaxial moment, are expected to be harder to estimate than beams, while braces, dominated by axial force, should be the easiest. Errors are likely to grow for the largest sections, where the data are sparse, and the remedy is to generate more building models rather than to tune the learner.

C. Design-level outcomes

TABLE IX. DESIGN COMPARISON FOR HELD-OUT BUILDINGS

Design	Concrete (m ³)	Steel (kg)	Cost	Verified
Original	[184.2]	[28,450]	[28,95,000]	[60.25 %]
Proposed	[161.5]	[24,180]	[24,82,000]	[97.50 %]
GA subset	[159.8]	[23,900]	[24,55,000]	[95.00 %]

Table IX is the central result. A reduction in cost has meaning only alongside the verification column, and one which disappears on re-analysis should be reported as such. The comparison with the genetic algorithm should state the number of STAAD.Pro runs each method consumed, because the claim of Hypothesis H4 concerns computing effort and not the final cost alone. A genetic algorithm that finds a cheaper design at several hundred times the analytical effort is a different kind of result from a model guided design that is slightly dearer but found in minutes, and the paper should state which of these has been demonstrated.

D. Explanation of model behaviour

A structural engineer should examine the SHAP summaries for both models. For beams, bending moment and span should dominate; for columns, axial force and the two moments should lead, in a manner consistent with (2). If a model relies

on features with no mechanical basis, such as an index or identifier, the pipeline should be reviewed for leakage.

E. Sensitivity to unit rates

The framework takes unit rates as an input, which makes a sensitivity analysis straightforward. Each of c_c , c_s and c_f is varied over a plausible band and the ranking of candidate sections is recomputed. The proportion of members whose selected section changes measures how far the conclusions depend on the market. A low proportion supports using the framework in firms whose rates differ from those assumed here.

IX. DISCUSSION

A. Interpretation

The framework fits best inside the design process as a screening and recommendation tool, placed between the first design iteration and the final design review. It does not replace either step. It points the engineer to the members worth investigating and to the material that may be removed, but it does not certify the design. This division of labour is appropriate and unavoidable given how the tools are built. The models learn from the output of code-based calculation, so they cannot be more authoritative than that calculation, and the final check has to be made in the design software.

B. Relation to existing work

Compared with metaheuristic approaches, the framework gives up the wide exploration of a global search in exchange for speed and reusability. A genetic algorithm can find a cheaper design for a single building given enough evaluations, but the boosted-tree models can answer new queries in milliseconds once trained. Compared with standalone prediction studies, the framework goes a step further by connecting the prediction to selection and verification. Compared with surrogate-assisted optimization, its models are trained across many buildings and then reused, rather than being trained inside each optimization run. These are differences of workflow, and the paper does not claim that boosted trees are intrinsically superior to other learners.

C. Practical implications

For a design office, the practical value lies in three places. Time is saved in early sizing, when many alternatives are cheap to consider. Consistency improves, since the same demands produce the same recommendation regardless of who is at the keyboard. And the accumulated analysis data, which are normally discarded, become an asset that improves the tool with each project. Academic institutions may also find the framework useful as a teaching aid, because it makes the relationship between demands and section size visible and testable.

D. Ethical and professional considerations

Structural safety is a professional responsibility. The application should therefore display a clear statement that its output is advisory, should log every recommendation with the model version that made it, and should refuse to process input that fails validation. Where a recommendation reduces a section, the engineer should be shown the evidence, namely the probability and the principal drivers, and should be asked to confirm. These measures reflect the principle in the programme outcomes of the department that computing professionals

commit to professional ethics and take responsibility for the consequences of their tools.

X. THREATS TO VALIDITY AND LIMITATIONS

A. Threats to validity

- *Construct validity.* Utilization from STAAD.Pro is a code-based proxy for safety. It inherits the assumptions of the analysis model and does not capture workmanship, material variability beyond the partial safety factors, or detailing errors.
- *Internal validity.* Leakage between rows is controlled by grouped splits, but features derived from the same analysis may still be correlated. The F1 and F2 comparison and the SHAP inspection are included to detect this.
- *External validity.* Generated buildings are regular frames. Results may not extend to irregular plans, transfer structures, flat slabs or buildings with shear walls.
- *Conclusion validity.* The number of independent buildings, not the number of members, determines statistical power. Tests are therefore run over folds and reported with that count.

B. Limitations

- The first-order assumption that member forces do not change when a section is changed is an approximation. Closed-loop verification reduces the resulting error but does not eliminate it.
- The framework covers sizing. It does not yet generate full reinforcement layouts or check serviceability limits other than through the utilization that STAAD.Pro reports.
- The section catalogue and unit rates are user-supplied and region specific. Results obtained with one catalogue do not transfer automatically to another.
- The models are trained on simulated buildings and need to be retrained or fine-tuned on real-world projects before they can be used in practice.
- The code basis is IS 456 : 2000 and the associated Indian standards. Other design codes will need their own datasets.

XI. FUTURE SCOPE

- Add embodied carbon as a second objective and report the cost and carbon trade-off as a Pareto front.
- Extend the framework to slabs, shear walls and foundations, and to buildings with irregular geometry.
- Replace the fixed first-order assumption in the selection layer with a learned model of force redistribution, reducing the number of verification iterations.
- Use active learning: whenever the closed loop finds a member on which the model was wrong, add it to the dataset with a higher weight.

- Quantify uncertainty with conformal prediction, which would give each recommendation a coverage guarantee under mild assumptions.
- Integrate directly with OpenSTAAD so that the application can read results and write sections back without an intermediate CSV file.
- Test the framework on anonymized data from real projects and compare against the decisions of practising engineers.

XII. CONCLUSION

This paper has documented a workflow in which the results of structural analyses performed in STAAD.Pro are formatted into a dataset, used to train XGBoost models, and turned into design recommendations that are then validated in the original software. The workflow addresses a gap in the literature. Evolutionary approaches, BIM-based scripts, and standalone machine-learning and surrogate models each tackle one aspect of design, but none of the sources reviewed here closes the loop from a commercial design application, through a boosted-tree learner, to a verified design.

The principal methodological points are the use of two models with an asymmetric treatment of safety errors, the separation of feature sets with and without utilization to avoid mistaking a threshold rule for learned understanding, the grouping of data by building model during validation, and the closed-loop check that allows only verified savings to be counted. The prototype STAAD.OPT shows that the workflow can be delivered as a small local application with a simple CSV interface that matches the output of the analysis package.

The experiments defined here, with their acceptance criteria fixed in advance, will show whether the approach reaches the standard needed for use as a screening aid. Whatever the outcome, the dataset construction procedure, the evaluation protocol and the open limitations described above are intended to be useful to later work on data-driven design of reinforced concrete buildings.

ACKNOWLEDGMENT

The authors thank the project guide and the Department of MCA for their guidance during the research project course.

REFERENCES

- [1] Bureau of Indian Standards, Plain and Reinforced Concrete: Code of Practice, IS 456, 4th revision, New Delhi, India, 2000.
- [2] Bureau of Indian Standards, Code of Practice for Design Loads (Other Than Earthquake) for Buildings and Structures, IS 875 (Parts 1 to 5), New Delhi, India, 1987.
- [3] Bureau of Indian Standards, Criteria for Earthquake Resistant Design of Structures: General Provisions and Buildings, IS 1893 (Part 1), New Delhi, India, 2016.
- [4] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 785–794.
- [5] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?," in Adv. Neural Inf. Process. Syst. 35 (Datasets and Benchmarks Track), 2022.
- [6] S. U. Pillai and D. Menon, Reinforced Concrete Design. New Delhi, India: Tata McGraw-Hill, 2003.
- [7] Bentley Systems, Incorporated, STAAD.Pro and OpenSTAAD Documentation, n.d.

- [8] R. Kicinger, T. Arciszewski, and K. De Jong, "Evolutionary computation and structural design: A survey of the state-of-the-art," *Comput. Struct.*, vol. 83, nos. 23–24, pp. 1943–1978, 2005.
- [9] A. Kaveh, *Applications of Metaheuristic Optimization Algorithms in Civil Engineering*. Cham, Switzerland: Springer, 2017.
- [10] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [11] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Inf. Fusion*, vol. 81, pp. 84–90, 2022.
- [12] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Adv. Neural Inf. Process. Syst.* 30, 2017, pp. 4765–4774.
- [13] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.
- [14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.
- [15] C. V. Camp, S. Pezeshk, and H. Hansson, "Flexural design of reinforced concrete frames using a genetic algorithm," *J. Struct. Eng.*, vol. 129, no. 1, pp. 105–115, 2003.
- [16] V. Govindaraj and J. V. Ramasamy, "Optimum detailed design of reinforced concrete continuous beams using genetic algorithms," *Comput. Struct.*, vol. 84, nos. 1–2, pp. 34–48, 2005.
- [17] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.
- [18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.
- [19] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Comput. Ind.*, vol. 41, no. 2, pp. 113–127, 2000.
- [20] M. Aldwaik and H. Adeli, "Advances in optimization of highrise building structures," *Struct. Multidiscip. Optim.*, vol. 50, no. 6, pp. 899–919, 2014.
- [21] H. Salehi and R. Burgueño, "Emerging artificial intelligence methods in structural engineering," *Eng. Struct.*, vol. 171, pp. 170–189, 2018.
- [22] H. Sun, H. V. Burton, and H. Huang, "Machine learning applications for building structural design and performance assessment: State-of-the-art review," *J. Build. Eng.*, vol. 33, Art. no. 101816, 2021.
- [23] H.-T. Thai, "Machine learning for structural engineering: A state-of-the-art review," *Structures*, vol. 38, pp. 448–491, 2022.
- [24] S. Mangalathu and J.-S. Jeon, "Classification of failure mode and prediction of shear strength for reinforced concrete beam-column joints using machine learning techniques," *Eng. Struct.*, vol. 160, pp. 85–94, 2018.
- [25] S. Mangalathu, H. Jang, S.-H. Hwang, and J.-S. Jeon, "Data-driven machine learning-based seismic failure mode identification of reinforced concrete shear walls," *Eng. Struct.*, vol. 208, Art. no. 110331, 2020.
- [26] D.-C. Feng, Z.-T. Liu, X.-D. Wang, Y. Chen, J.-Q. Chang, D.-F. Wei, and Z.-M. Jiang, "Machine learning-based compressive strength prediction for concrete: An adaptive boosting approach," *Constr. Build. Mater.*, vol. 230, Art. no. 117000, 2020.
- [27] H. T. Mai, J. Kang, and J. Lee, "A machine learning-based surrogate model for optimization of truss structures with geometrically nonlinear behavior," *Finite Elem. Anal. Des.*, vol. 196, Art. no. 103572, 2021.
- [28] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
- [29] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York, NY, USA: Springer, 2009.
- [30] Bureau of Indian Standards, *Ductile Design and Detailing of Reinforced Concrete Structures Subjected to Seismic Forces: Code of Practice*, IS 13920, New Delhi, India, 2016.
- [31] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 2623–2631.
- [32] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.
- [33] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

APPENDIX A. REPRODUCIBILITY CHECKLIST

- Record the STAAD.Pro version, the design code edition and the load combinations used for every generated model.
- Store the generation script, random seeds and the list of building identifiers assigned to each cross-validation fold.
- Pin the versions of Python, XGBoost, scikit-learn and the web framework in a requirements file.
- Save the trained models, calibration objects, selected thresholds and the section catalogue with unit rates.
- Log, for every recommendation, the input row, model version, probability and the verification outcome.