Journal of High School Science

Leveraging surrogate modeling using synthetic data for energy efficient decision-making in residential buildings

Kaur M[1], Gupta D[2]

Submitted: July 20, 2024, Revised: version 1, September 3, 2024, version 2, September 18, 2024, version 3, September 29, 2024, version 4, October 5, 2024
Accepted: October 6, 2024

Abstract

Thermal building simulations play an important role in achieving energy efficiency, resulting in reduced carbon emissions and operational costs. However, such simulations are input-intensive and may need to be performed by modeling experts. This becomes a bottleneck in the building design process, especially for small residential projects, where there are limitations in terms of time and access to thermal modeling experts and tools. To address this issue, we explored the viability of replacing the simulation modeling approach with Machine learning (ML) models. We accessed and evaluated the suitability, performance and precision of ML models trained on data generated by building thermal simulations. Our study focused on a 2-storey residential building model simulated using the modeling software EnergyPlus™, representing a typical small residential project. We trained and tested multiple classical ML models and Deep learning models to predict the Energy performance index (EPI) of the building. Sensitivity analysis was also performed on the data to assess the effect of various parameters on the EPI and to identify which factors contributed the most to the accuracy of the model. After extensive experimentation and evaluation, we found that the Artificial Neural networks (ANN) was the most accurate. Our model was able to make predictions faster than traditional simulation models without significant loss of accuracy, highlighting the potential for using such surrogate models in place of input-intensive and computation-heavy simulations during the early design stages. This may help in monitoring and exploring multiple design scenarios, streamlining the assessment of building efficiency, non-intrusive load monitoring, and facilitating informed decision-making for policymakers.

Keywords

Building simulation, Energy Performance Index, Thermal simulation of Buildings, Surrogate energy model, Artificial Neural Network, Machine Learning, EnergyPlus™

[1]Corresponding author: Mannat Kaur, Delhi Public School, RK Puram, New Delhi 110066, India. mannatkaur1115@gmail.com

[2]Debayan Gupta, Department of Computer Science, Ashoka University, Plot 2, Rajiv Gandhi Education City, Post Office Rai, Kundli, NCR, Rai, Sonipat District, Haryana,131029, India. debayan.gupta@ashoka.edu.in

1 Introduction

India is one of the fastest growing economies in the world and has undergone rapid urbanization. This has led to a significant increase in the demand for residential units. Market analysis indicates a large year-on-year growth in the sales of residential units post-Covid. This heightened demand and growth, coupled with large-scale electrification and increased penetration of consumer appliances has made the residential sector one of the biggest consumers of electricity in the building sector. It is estimated that residential energy consumption will triple (246 TWh/y to 748 TWh/y) between 2017 and 2030 (NITI Aayog, 2015). With buildings exerting such significant pressure on the natural environment, energy optimization in the residential sector is an important aspect which would also contribute to India's Intended Nationally Determined Contributions (INDCs) aimed at reducing the intensity of the emissions linked to the Gross domestic product (GDP). Thermal simulations of Buildings can play a pivotal role in potential homeowners making informed decisions for energy efficiency, helping achieve energy optimization in the residential building sector. However, the entire process of thermal simulation of Buildings is input-intensive, time-consuming and computation-heavy and requires a fair amount of technical knowledge of the field.

Mahmoud et al. (1) found that lack of awareness and cost were barriers in the adoption of building thermal simulation during the early stage of design. Other technical barriers identified during the research included a poor, non-intuitive user interface and challenges related to data input, output processing and interpretation. Pang et al. (2) found that achieving optimization through building thermal simulation required a large number of inputs to include various building parameters, which is impractical as it becomes time and resource intensive.

Parametric modeling has proven to be an invaluable tool for optimizing and enhancing building energy performance. By systematically altering individual parameters, this approach allows fine-tuning and identifying optimal solutions through numerous simulations. However, the extensive number of simulation runs required can be time-consuming. Additionally, a significant drawback of this method is its inability to account for the interdependencies among various parameters (2). Data-driven models offer a solution that can reduce the computational cost associated with optimization and simulation tool coupling. These models are also referred to as "Surrogate models" or "Meta-models". A physics-based model may be used to generate large datasets for training an ANN which serves as the surrogate model (3).

Using surrogate models for optimization is computationally more efficient compared to physics-based models. However, it is a challenging process as it requires a large amount of data encompassing a wide range of desired parameters. Li et al. (4) developed various simulation-based optimization strategies and algorithms and performed a thorough analysis of their accuracy which included the GenOpt method and the ANN method. This analysis was performed within the context of a case study of a simple building energy model. Magnier et al. (5) integrated an ANN with a genetic algorithm aimed at minimizing both the energy consumption and implementation costs for residential buildings. Their approach focused on optimizing Heating, Ventilation and Air conditioning (HVAC) and building envelope parameters to achieve optimal energy

efficiency. Gossard et al. (6) developed an ANN-based prediction model for annual energy consumption forecasting and a thermal comfort index with a multi objective Genetic algorithm. A multi-objective optimization model combining an ANN with Genetic Algorithm was developed by Asadi et al. (7) to identify effective strategies for a building energy retrofit to minimize building energy consumption and implementation costs while simultaneously maximizing thermal comfort within the retrofitted buildings.

Small residential projects often lack access to experienced energy modelers during the early design stages, when there is the greatest flexibility to incorporate energy-saving measures. A pre-trained, rapidly implementable ML model can drive sustainable, low-carbon design decisions early in the process, offering maximum benefits. This paper focuses on developing machine learning algorithms and an ANN-based surrogate model that can serve as a proxy for building thermal simulation models, predicting the Energy Performance Index (EPI) with minimal input. Parametric runs from EnergyPlus™, the building thermal simulation software for residential buildings were used to train the ANN model. The model's accuracy was validated by comparing its predictions to the original EnergyPlus™ simulation results. Additionally, the study compared classical ML models and ANN models based on their accuracy. By leveraging these models and a comprehensive dataset, the paper aimed to identify the most effective techniques for accurately predicting energy consumption in residential buildings while significantly reducing computational resources.

2 Energy simulation model

A 3D building thermal modeling approach was adopted to generate the synthetic dataset, which was then used to train and develop the ANN-based surrogate model. Building energy simulation models are advanced tools (eQuest®, EnergyPlus™, Openstudio®, DesignBuilder® etc.) used to simulate building energy consumption and estimate their Energy Performance Index (EPI). A 3D model of the building is created, and various inputs that affect energy consumption—such as heating, cooling, lighting, ventilation, and other operational needs—are input into the model. Key inputs include building geometry, orientation, construction materials, envelope characteristics, internal loads, occupancy patterns, lighting and equipment loads, HVAC system types, and standard weather data.

The simulation model calculates the thermal load and the operation of air conditioning, lighting, and equipment based on usage patterns, hourly weather data, and control mechanisms. It also considers interactions between different systems; for instance, heat generated by lighting and equipment may reduce heating loads in winter but increase cooling demands in summer. These interactions are vital for producing accurate energy consumption estimates.

The simulation is typically conducted on an hourly basis over an entire year, calculating energy consumption for each hour while accounting for the dynamic interactions between building systems and the external environment.

2.1 Residential model details

A 3D building thermal model (Figure 1) of a double-story residential building was prepared. Each household was simulated with two bedrooms, a living room, a dining room, a kitchen and two washrooms. These spaces

collectively occupied ~ 72 m2 of carpet area. The purpose of this configuration was to represent a typical residential building layout commonly found in urban areas.

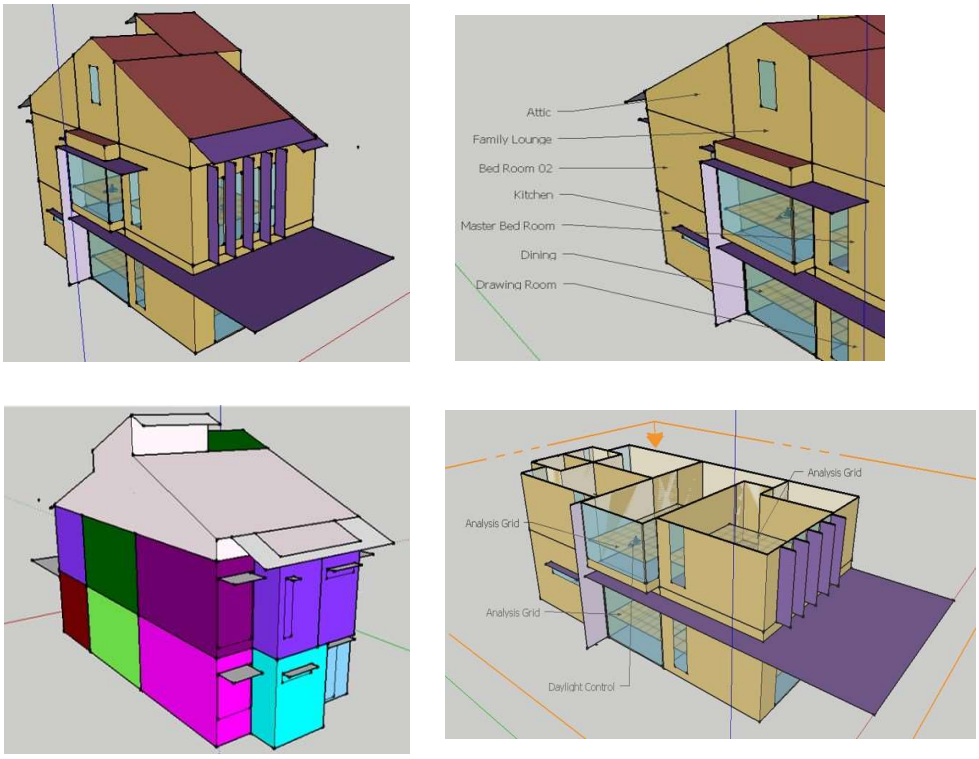


Figure 1. Simulation Model: 3D view of the model (top left), 3D view with space allocation details (top right), Thermal zoning of the unit (bottom left), Sectional view of the model (bottom right).

The occupancy of the building was based on a standard urban family composition. It was assumed that each household is occupied by four individuals, comprising a husband, a wife, and two children. The assumption of a four-member family allowed for a reasonable representation of average occupancy patterns and the corresponding heat generation within the building. assumed that within each household, one individual was a working individual, one was a homemaker and the remaining two were school-going children. Consequently, this assumption implied that the household would be either partially or fully occupied at all times. This assumption accounted for the varying activity levels and heat generation within the building throughout the day.

Basic amenities commonly found in residential buildings were considered in the model. These included lighting fixtures, ceiling fans and home appliances such as refrigerators, televisions and other electronic devices. These amenities are essential for the comfort of the occupants and daily activities. Additionally, the family composition within each household also influenced the occupancy patterns. It was

2.2 Parameter definition and context

To evaluate the energy performance of the building under different conditions, we considered a range of parameters that affected both the thermal efficiency and overall energy consumption. These parameters spanned various aspects of the building's design, mechanical systems, and environmental context. The Wall U-value measures the thermal

insulation capacity of the building's walls, affecting heat gain and loss through the external walls. The Roof U-value assesses the insulation performance of the roof, which directly influences the building's heat gain from the roof. The Types of Glazing presents the different glazing materials that affect the heat transfer and solar heat gain through the windows. The Shading Strategies (such as overhangs or louvers) help reduce direct solar heat gain through windows. The efficiency of Air Conditioning systems directly influences energy consumption. More efficient systems provide the same cooling effect with less energy, reducing overall energy use. The Window-to-Wall Ratio (WWR) represents the ratio of window area to total wall area and influences heat gain and heat loss through non-opaque building elements such as glass. The Building Orientation dictates how much solar radiation each façade is exposed to throughout the day. Proper orientation can help minimize or maximize heat gain on sun-exposed facades. Integrating Natural Ventilation when outdoor conditions are favorable can significantly reduce the need for mechanical cooling systems. The building's context determines its exposure to external elements (standalone, exposed on two sides, or three sides open), which affects its vulnerability to heat gain or loss. Buildings with more sides exposed are more susceptible to external temperature variations, which, in turn, affect energy consumption. Different climate types (e.g., hot, temperate, or cold) determine the building's thermal performance.

2.3 Synthetic data sets

The thermal simulation model in EnergyPlus™ was used, and a scripting process was devised to modify one parameter at a time before executing the simulation. By altering one parameter, multiple EnergyPlus™ simulation runs were performed to calculate the building's EPI metric. The output EPI values for each case were extracted into a CSV file. Using this scripting approach, we generated and simulated 116,640 parametric runs. Table 1 indicates the values for which parametric runs were conducted and the extract from the dataset generated through parametric building thermal simulation modeling, illustrating key input variables and their corresponding Energy Performance Index (EPI) outputs.

3 Surrogate model

Artificial Neural Networks (ANNs) represent a modeling method that is inspired by neurons constituting the human brain. They consist of interconnected nodes, or neurons that process information and recognize patterns. They are mapping tools loosely modeled based on how neurons in the human brain behave. The architecture of ANNs consists of different layers - an input layer, hidden layers and an output layer. Each layer is further divided into several blocks called nodes that perform computations to extract features and patterns from the input data. To facilitate complex computations, each layer employs a unique activation function. ANNs are a subset of machine learning and are particularly effective in handling complex tasks such as image recognition, natural language processing, and predictive analytics. Various architectures exist, including Feedforward Multi-Layer Perceptron (MLP), Recurrent Neural Network (RNN), Generative Adversarial Networks (GANs) and Radial Basis Function networks.

The learning mechanism in artificial neural networks (ANNs) revolves around training, where the network fine-tunes its weights and biases based on the input data. This training involves an error reduction technique known as backpropagation, where the network calculates the error between the predicted output and the

actual output, then propagates this error backward through the network to update the weights accordingly. This iterative process continues until the network achieves a satisfactory level of accuracy. In this paper, we designed and defined the structure of classical ML and ANN models that were trained to replace the computational and time-intensive building simulation model.

Table 1. Individual parameters used to create training and validation data and snapshot of the generated dataset.

Parameter	Value	Description
Wall Type (U-value: W/sqmK)	2.256	Plastered Brick Wall
	0.720	Plastered AAC Block Wall
	0.719	Plastered Brick wall with Insulation (50 mm)
	0.463	Plastered Brick wall with Insulation (100 mm)
Roof Type (U-value: W/sqmK)	2.865	RCC Slab laid with Mud Phuska
	0.869	RCC Slab laid with Insulation (50mm)
	0.492	RCC Slab laid with Insulation (100mm)
Glass (U-value: W/sqmK) / SHGC	5.591/0.58	Single Glazing
	2.502/0.32	Double Glazing
	1.019/0.18	Triple Glazing
Window to Wall Ratio	0.2	Reasonably Sized Windows
	0.3	Large Windows
	0.45	Larger Windows
Shade	0	No Shading
	0.5	50% Shaded Windows
	1.0	100% Shaded Windows
Orientation (degree)	0	North Facing
	90	East Facing
	180	South Facing
	270	West Facing
AC (Coefficient of Performance)	2.4	1 Star AC
	3.0	3 Star AC
	3.4	5 Star AC
Natural Ventilation	-	Yes, No
Building Configuration	-	Stand-Alone
	-	Two-side Open
	-	Three-side Open
Climate	-	Hot and Dry
	-	Temperate
	-	Warm and Humid
	-	Composite
	-	Cold

1	Climate	Wall U-va	Roof U-va	Glass U-	Glass SH	WWR	Shade	Orientatio	COP	NV	BC
0	Composite	2.256	2.865	5.591	0.58	0.2	0	0	2.4	No	Stand-Alone
1	Composite	2.256	2.865	5.591	0.58	0.2	0	0	3	No	Stand-Alone
2	Composite	2.256	2.865	5.591	0.58	0.2	0	0	3.4	No	Stand-Alone
3	Composite	2.256	2.865	5.591	0.58	0.2	0	90	2.4	No	Stand-Alone
4	Composite	2.256	2.865	5.591	0.58	0.2	0	90	3	No	Stand-Alone
5	Composite	2.256	2.865	5.591	0.58	0.2	0	90	3.4	No	Stand-Alone
6	Composite	2.256	2.865	5.591	0.58	0.2	0	180	2.4	No	Stand-Alone
7	Composite	2.256	2.865	5.591	0.58	0.2	0	180	3	No	Stand-Alone
8	Composite	2.256	2.865	5.591	0.58	0.2	0	180	3.4	No	Stand-Alone
9	Composite	2.256	2.865	5.591	0.58	0.2	0	270	2.4	No	Stand-Alone
10	Composite	2.256	2.865	5.591	0.58	0.2	0	270	3	No	Stand-Alone

3.1 Data preprocessing and exploration

Data preprocessing is an important step in all the models which involves cleaning and

transforming the input data to ensure it is suitable for training. 1) Categorical data values: Categorical features, such as climate type, natural ventilation (yes/no), and building configuration (e.g., standalone, two sides exposed), were processed using categorical encoding. These features were grouped by their respective categories and encoded using label encoding. For each category, the number of occurrences of the target was computed, and the probability of each target within a category was calculated. This encoding method preserved the categorical distinctions between features while ensuring the model could interpret and utilize them effectively. 2) Numeric data values: Continuous numerical features, such as Wall U-value, Roof U-value, and Glass U-value/SHGC, window-to-wall ratios, Orientation, and AC COP were scaled using robust standardization. This method minimized the influence of outliers and standardized the features to ensure consistent variance across all variables. By scaling the data, we ensured fair comparison between different features, improving the performance and reliability of the models. 3) Preprocessing pipeline: A pre-processing pipeline was constructed to process categorical and numeric features. This pipeline applied different transformations to different subsets of features within the dataset. A function was implemented to streamline the process of extraction of transformed feature names, providing a better understanding of the structure of the transformed data after applying various pre-processing steps. 4) Train-Test Split: To assess the model generalization on unseen data, a portion of the initial data was not included in the intermediate models and was reserved for validation purposes. This was done to verify the accuracy and reliability of the final model when applied to new, previously unseen data. The dataset was split into training and testing sets (80% training, 20% testing) with the training dataset comprising 93,312 data points and the test dataset comprising 23,328 data points. This split is a standard practice in ML algorithms which is used to strike a balance between having enough data for training and ensuring a robust evaluation on the test set. Reducing the proportion of training data can hinder the model's ability to learn important patterns, while increasing it may leave insufficient data for testing, making it difficult to accurately evaluate the model's performance on new scenarios.

3.2 Classical ML algorithms

There are several types of ML algorithms each suited for different types of tasks and data. Some of the ML algorithms used in the study were XGBoost, Decision Tree, Lasso Regression, ElasticNet (Zou and Hastie(8)), LinearSVR (Klopfenstein and Vaiter(9)), Linear Regression and Random Forest Regressor (Breiman(10)). XGBoost (Chen and Guestrin(11)). is a powerful gradient boosting algorithm known for its speed and accuracy, particularly effective with large datasets. Random Forest Regressor is an ensemble learning method that combines multiple decision trees to improve accuracy and control overfitting, effectively capturing nonlinear relationships and feature interactions. Decision Tree models are non-parametric and split data into subsets based on feature values, making them easy to interpret and visualize. Lasso Regression is a linear model with L1 regularization that shrinks coefficients to select important features, preventing overfitting. ElasticNet combines L1 and L2 regularization, balancing between Lasso and Ridge regression to handle complex datasets and multicollinearity. LinearSVR is a scalable and robust support vector regression model, effective for high dimensional spaces. Linear Regression assumes a linear relationship

between input features and the target variable, serving as a simple and interpretable baseline model.

Hyperparameter tuning was performed to optimize model performance, reduce overfitting and ensure that the model generalized well to unseen data. This involved systematically searching for the best combination of hyperparameters, which are settings not learned during training but rather, set before training begins.

A comprehensive grid search with 5-fold cross-validation was employed to select the optimal combination of parameters for each algorithm. Cross-validation was used to ensure the robustness of the results by splitting the data into multiple subsets and performing training and evaluation iteratively. A logarithmic transformation was applied to the target variable to increase its distributive properties and an exponential transformation was applied for inverse prediction, enabling the model to work effectively in its predictive capacity. A predefined hyperparameter grid specified the search space and search strategy for optimizing model parameters. The refit parameter was configured to use Negative Mean Absolute Error as the performance metric indicating that the model would be refitted using the best hyperparameters identified during the grid search. The parallel processing (n_jobs) parameter was set to -1 which enabled the utilization of all available CPU cores for parallel computation thereby accelerating the computational process. After implementing the baseline model with the specified parameters, various other standard models were utilized to examine different regression approaches and evaluate their performance relative to the baseline model.

3.3 Surrogate model development

This section explores the implementation of Surrogate models using opensource libraries. These frameworks provide flexible interfaces for developing neural network architectures and training deep learning models. They also enable automatic differentiation and GPU acceleration, making them ideal for deep learning model training.

To enhance the model's ability to learn patterns and capture the non-linear relationships in the data, several feature engineering techniques were applied. The Surrogate model development consisted of 1) Normalization, 2) ANN model development and 3) Training, validation, and testing. The dataset was first preprocessed through a combination of label encoding for categorical variables and standard scaling for continuous features. Standard scaling transformed these features to have zero mean and unit variance, an essential step that ensured that features with different orders-of-magnitude did not disproportionately influence the model's weights.

This shifted the data distribution to have a mean of zero placing the feature values around the origin of the coordinate system. Subsequently, the centered feature values were divided by the standard deviation of the feature. This normalized the feature values and ensured they possessed a unit variance, making them comparable on the same scale. It was essential to ensure that all input features comprised of ranges of similar orders-of-magnitude because widely spread values cause large weight value variations and result in an unstable model. The standard score (z) of sample x was calculated as follows, where μ is the mean of the training samples, and σ is the standard deviation of the training samples.

$$z = \frac{x-\mu}{\sigma} \quad (1)$$

The TensorFlow framework was used for implementing and training the ANN model. The neural network architecture was a feed-forward design with three fully connected hidden layers. The ANN was designed with varying numbers of neurons across the input, hidden, and output layers and the number of neurons in the input layer corresponded to the number of input features in the training dataset.

The architecture consisted of four hidden layers with a progressively decreasing number of units, creating a funnel-like structure to distill information from the input layer. The first hidden layer contained 512 units and used the Rectified Linear Unit (ReLU) activation function, which introduced non-linearity, allowing the model to capture complex relationships between features and target outputs. In the exploratory phase of model tuning, extensive experimentation was performed with different hyperparameters to optimize the performance metrics. The number of neurons in each hidden layer was varied across multiple configurations, starting from shallow architectures with fewer neurons to deeper architectures with larger number of neurons, in order to assess the trade-off between model complexity and generalization. Various activation functions were also evaluated during the exploratory analysis phase to obtain the best output, such as sigmoid, Tanh (hyperbolic tangent) function, and ReLU (Rectified Linear Unit), among others. Additionally, L2 regularization with a penalty factor of 0.01 was applied to this layer. Different values were tested to identify the optimal regularization strength that achieved a balance between fitting the training data well and generalizing to unseen data effectively. L2 regularization penalizes large weights by adding the squared sum of weights to the loss function, thus reducing the risk of overfitting.

In the second hidden layer, comprising 256 units, ReLU activation and L2 regularization continued to be utilized to ensure that the model remained capable of processing non-linear patterns while controlling the complexity of the weights. The third hidden layer reduced the units to 128 and applied similar activation and regularization strategies. This gradual decrease in the number of units across the layers facilitated the model's ability to effectively compress the learned representations, refining high-level abstractions from the raw input features. The fourth and final hidden layer contained 64 units, continuing the use of ReLU and L2 regularization to maintain the model’s capacity to learn useful patterns while mitigating the risk of overfitting.

The output layer consisted of a single unit with a linear activation function, indicating that the model performed a linear regression and predicted a continuous numeric value. The inclusion of standard scaling ensured that all features contributed equally during the learning process, preventing dominance by any feature due to its different order-of-magnitude, while L2 Regularization generalized the model by preventing over-reliance on any particular feature or noise within the training data. Different values were tested to identify the optimal regularization strength that achieved a balance between fitting the training data well and generalizing to unseen data effectively. The model was trained using the Adagrad (Adaptive Gradient Algorithm) optimizer with a learning rate of 0.01. Various Optimizers, including Adam, RMSProp, SGD were evaluated during initial trials. A range of learning rates was also explored to determine the most suitable value. A lower learning rate led to slow convergence and did not improve the validation accuracy significantly, while higher rates caused erratic behaviors and instability in the learning

process. The batch size was set to 128 and the training process was conducted for a maximum of 1000 epochs. The early stopping callback was included to monitor the validation loss and stop training if the validation loss did not decrease for 3 consecutive epochs. The weights and biases were updated through an iterative process to minimize the MAE and obtain an accurate model. After training, the code switched the model to evaluation mode and used it to predict the output for the test set. The MAE, MAPE and coefficient of determination (R^2) were calculated between the predicted values and the ground truth, providing a measure of the model's performance. By systematically adjusting the number of neurons, hidden layers and regularization, while experimenting with multiple optimizers, the final model architecture and hyperparameters were selected based on their ability to minimize these error metrics.

4 Results & discussion

The outputs of both ANN and classical ML algorithms were evaluated by comparing them to the corresponding building energy model outputs. The performance was compared using (R^2), square of the linear correlation coefficient, Mean Absolute error and Mean absolute percentage error. The Coefficient of determination (R^2) assesses how well the model captures the relationship between the variables. A higher value of R^2 closer to 1 indicates a better fit, while a value closer to 0 suggests that the model does not explain much of the variance in the data. The Mean absolute error (MAE) is a measure of the average difference between the predicted and actual values. Compared to other error metrics such as Mean Squared Error (MSE), MAE gives equal weight to each prediction error without squaring the differences which makes it less sensitive to outliers in the data. The MAE is computed using the equation, where, y_i = predicted values, x_i = true values, and n = total number of datapoints

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i| \qquad (2)$$

The Mean absolute percentage error (MAPE) is used to measure the average magnitude of error produced by a model and to analyze how far off predictions are on average. It is used to define the accuracy of a ML algorithm. The MAPE is computed using the equation, where, A_t = actual values, F_t = forecast values, and n = total number of datapoints.

$$MAPE = \frac{1}{n}\sum_{t=1}^{100}\left|\frac{A_t - F_t}{A_t}\right| \qquad (3)$$

4.1 Classic ML models performance

Classical ML models such as Decision Tree, Elastic Net, LinearSVR, Linear Regression, Lasso Regression were evaluated, but only the results from the XGBoost and Random Forest Regressor are discussed in detail due to their better performance. This indicates that the dataset contained non-linear relationships and complex feature interactions that these simpler models could not capture. These tree-based models demonstrated robustness to noise and outliers, highlighting their suitability for the dataset.

Table 2 MAE, MAPE and R^2 values for the different models trained.

Model	MAE	R^2	MAPE
XG Boost	0.792077	0.997338	1.0270621
Random Forest Regressor	1.092245	0.991543	1.889381
Decision Tree	1.431711	0.993903	1.436687
ElasticNet	8.200106	0.731285	10.270413
LinearSVR	8.253877	0.725827	10.315151
Linear Regression	8.212571	0.730301	10.260597
Lasso Regression	8.170789	0.751918	10.657131

Figure 2(a) and 2(b) present the performance of the XGBoost and Random Forest models respectively which were the most accurate among all the classical models evaluated. Each point on the graph represents an individual sample, with its horizontal position representing the actual EPI value and its vertical position indicating the model's prediction. Both models showed scatter points clustered closely around the identity line (y = x), indicating overall accuracy. Both models performed well for mid-range values (60-130), where data points were densely concentrated.

The XGBoost model consistently aligned with the identity line across all value ranges, indicating its robust ability to generalize to both lower and higher EPI values. Slight deviations from the line were visible at the higher end, but overall, the model demonstrated high predictive accuracy.

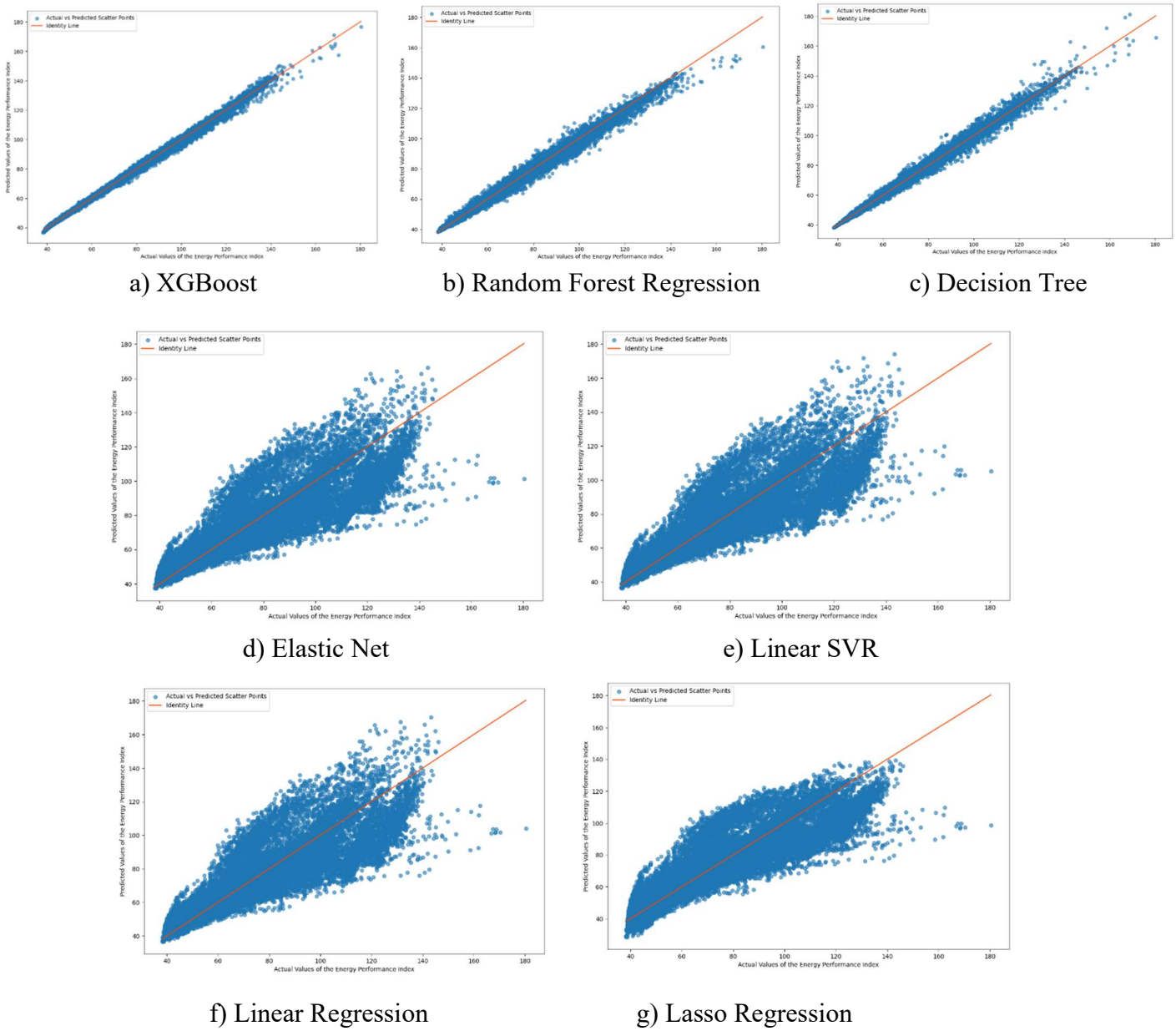
In contrast, while the Randomforest Regressor also showed good performance within this range, it underpredicted for values above 140 compared to XGBoost. This may have been due to insufficient training data points for these greater values.

The distribution plots presented in Figure 3(a) and 3(b) provide a visual representation of the EPI values and their corresponding frequencies as predicted by the XGBoost and Randomforest regressors. The x-axis represents the EPI values, while the y-axis shows the count of instances for each EPI value range. Both models exhibited a similar pattern to the actual data, indicating a fair degree of alignment. The histogram bars, representing the count of EPI values, peaked ~ 60-70 and tapered off symmetrically on either side, though the tail was slightly right skewed. Both models demonstrated strong predictive performance, with the XGBoost model showing slightly better alignment with the actual values, particularly in the higher EPI ranges.

The residual plots shown in Figure 4(a) and 4(b) illustrate the relationship between predicted EPI values on the x-axis and their corresponding residuals on the y-axis. The majority of residuals, which reflect the differences between predicted and actual values, were clustered around the zero line, indicating that the model's predictions were generally accurate, unbiased and possessed constant variance. There were no obvious patterns or systematic deviations visible in the plot, suggesting that the models were fitting without bias. However, there was some heteroscedasticity (non-constant variance) visible in both plots, particularly for larger predicted values (above ~120) which could be observed through the funnel shape, where residuals spread wider as predicted values increase.

The error histograms shown in Figure 5(a) and 5(b) illustrates the distribution of prediction errors from XGBoost and Randomforest Regressor. Both the graphs exhibited a pronounced peak around zero, indicating that most prediction errors were minimal. The XGBoost histogram was fairly symmetrical

around zero, suggesting that the model did not significant bias towards overestimation or underestimation. It presented with a more symmetrical error distribution and a narrower error range (-2 to 6), indicating tighter error margins. This symmetry and narrow range implied that the XGBoost model was better at processing outliers and extreme values and performed well for the majority of predictions, though it still presented with occasional larger positive errors, as seen from the slight right skew in the tail.



a) XGBoost b) Random Forest Regression c) Decision Tree

d) Elastic Net e) Linear SVR

f) Linear Regression g) Lasso Regression

Figure 2. Comparison of actual versus predicted values of Energy Performance Index using different Classical regression models.

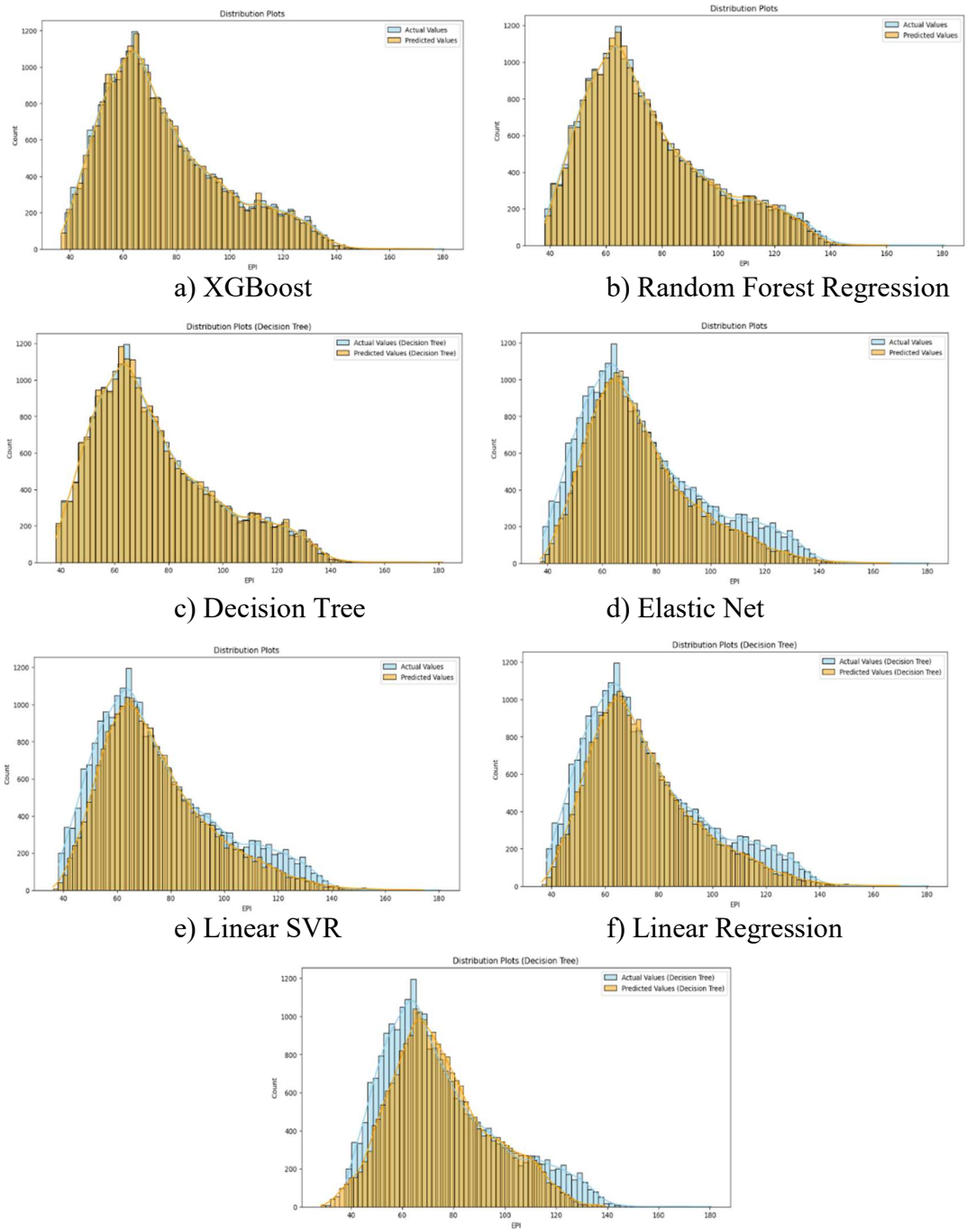


Figure 3. Distribution plot of frequency counts for actual versus predicted Environmental Performance Index values.

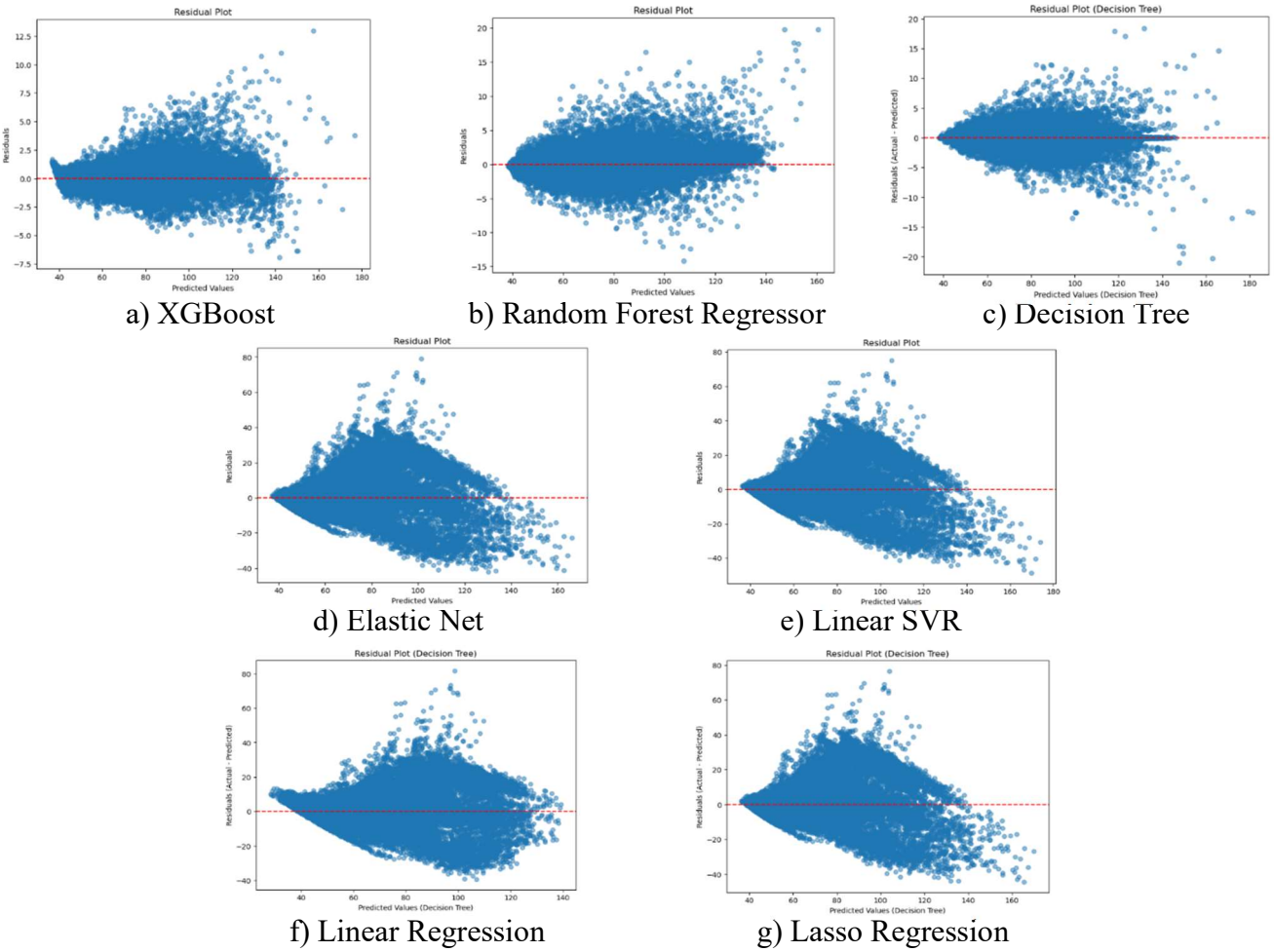
In contrast, the Randomforest Regressor histogram was slightly right-skewed, indicating a minor tendency towards underestimation. The error range for the Random Forest model was broader (-15 to 20), suggesting that it occasionally produced larger errors. While still accurate, this broader range implied that the Random Forest model was more sensitive to variations in the data, which could be advantageous in scenarios requiring a wider range of behaviors to be captured. The y-axis of both the histograms shows the frequency of errors, with the highest frequency occurring at the center for both the models. The frequency decreased rapidly as the errors moved away from zero, indicating that large errors were rare.

This common pattern underscored that both models generally made small errors, but the XGBoost model demonstrated a more consistent performance with fewer extreme errors.

The hyperparameter tuning results for the Random Forest model identified the following optimal settings: Bootstrap: False, Max Depth: 25, Max Features: sqrt, Min Samples Leaf: 1, and Min Samples Split: 2.

The model showed a clear trend where increasing the maximum depth of the decision trees from 10 to 20 led to a significant reduction in the Mean Absolute Error (MAE) for both the training and test sets. Beyond a depth of 20, the MAE scores stabilized, indicating that a maximum depth of 25 was the most effective balance between model complexity and performance. The performance of the model was not particularly sensitive to the number of features considered for each split, with both sqrt and log 2 yielding similar results.

For the Min Samples Leaf parameter, smaller values (such as 1 or 2) were found to contribute to better model accuracy, as they resulted in lower MAE scores. An increase in this parameter from 1 to 10 resulted in a gradual increase in the MAE for both the training and test sets. Similarly, for the Min Samples Split parameter, lower values (~ 2 to 5) also improved model performance. A split was only performed if there were at least two samples available for each internal node. These hyperparameters were identified as the most optimal in terms of performance, resulting in the best MAE, MAPE and R^2 scores for the Random Forest Regressor.



a) XGBoost b) Random Forest Regressor c) Decision Tree

d) Elastic Net e) Linear SVR

f) Linear Regression g) Lasso Regression

Figure 4. Residuals between true and predicted values for Energy Performance Index.

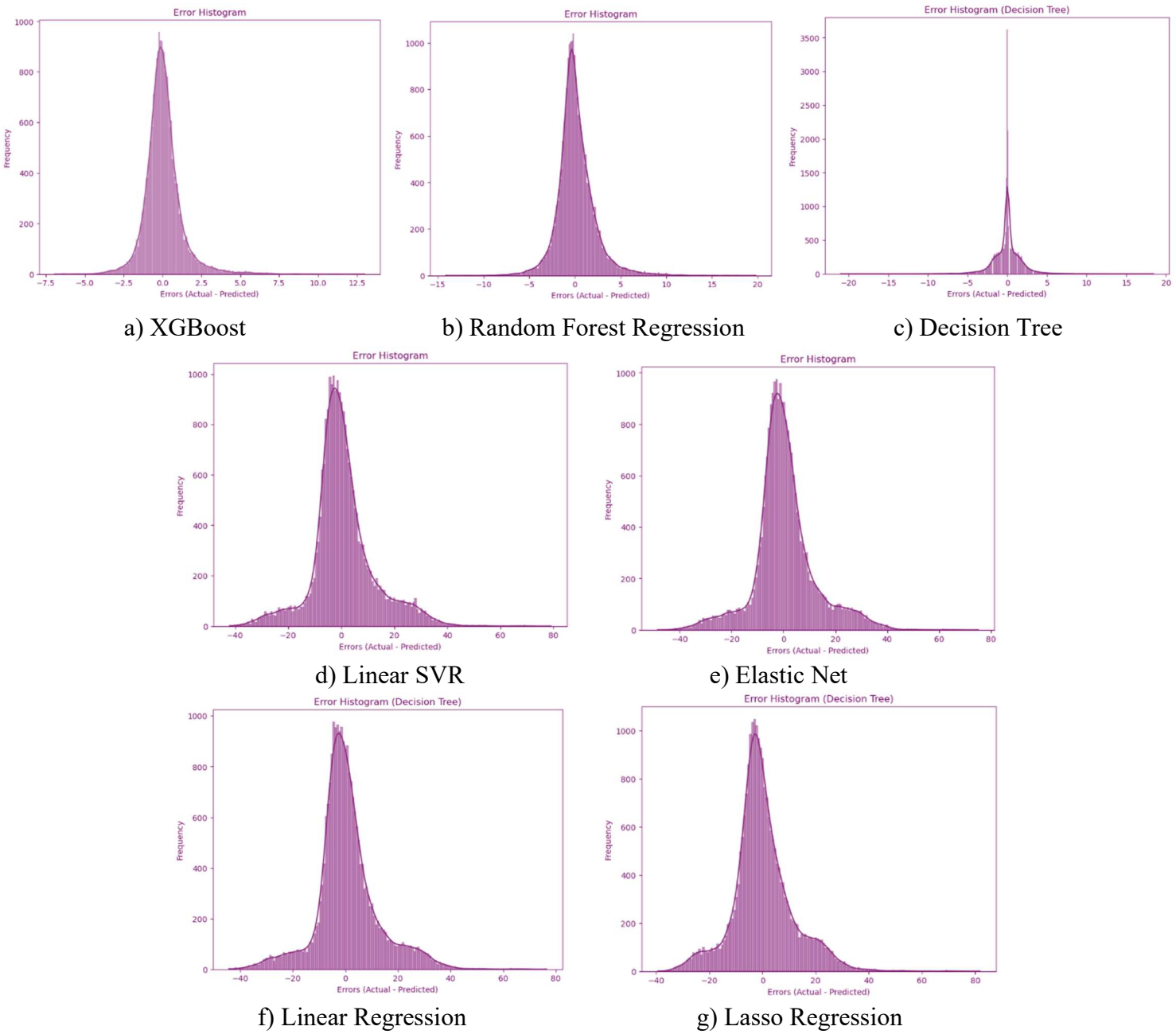


Figure 5. Error Histogram: Frequency Distribution of Prediction Errors.

4.2 ANN model performance

Among all the models tried, the ANN achieved the lowest MAE, MAPE and greatest regression coefficient. Figure 6 illustrates the training and validation loss of a neural network model plotted against the number of epochs, starting from epoch two. The training loss and validation loss both exhibited a steep decline during the initial epochs, indicating that the model was rapidly learning and improving its performance on both the training and validation datasets. The sharp decrease in loss values suggested that the model was effectively minimizing the error through the optimization process.

The loss values began to stabilize and plateau ~ 5 to 10 epochs, indicating that the rate of improvement slowed as the model converged. From that point onward, the curves flattened

with minimal additional reduction in loss and by the final epoch the model achieved a loss close to 0, demonstrating effective convergence.

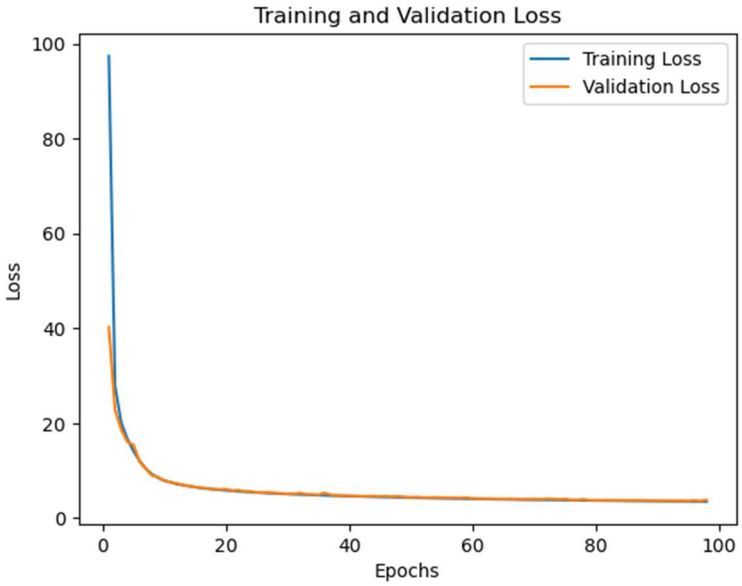


Figure 6. Artificial Neural Network loss values over successive training epochs.

The close alignment of the training and validation loss curves throughout the process suggested that the model generalized well, as there was no significant divergence between the two curves, indicating minimal overfitting. The ANN also achieved a regression correlation coefficient close to 1, signifying a strong relationship between the predicted outputs and the actual target values.

Table 3. MAE, MAPE and R^2 values for different models trained.

Model	MAE	R^2	MAPE
ANN	0.702303	0.997819	0.0108767
XG Boost	0.792077	0.997338	1.0270621
Random Forest Regressor	1.092245	0.991543	1.889381
Decision Tree	1.431711	0.993903	1.436687

4.3 ANN model interpolation performance

In this section, we evaluated the interpolation capabilities of the ANN model by testing its ability to predict intermediate values that were entirely absent from the training dataset. This evaluation aimed to assess the model's generalization ability and robustness when applied to unseen data points, particularly values between the known training inputs. Permutation Feature Importance technique was used to assess the significance of individual features in predicting the target variable. (Figure 7) demonstrates the permutation feature importance for the model, highlighting the relative contribution of each feature to the Energy Performance Index (EPI) prediction. Features with larger negative values are more influential, while those with smaller or near-zero values have a lesser impact on the model's predictions.

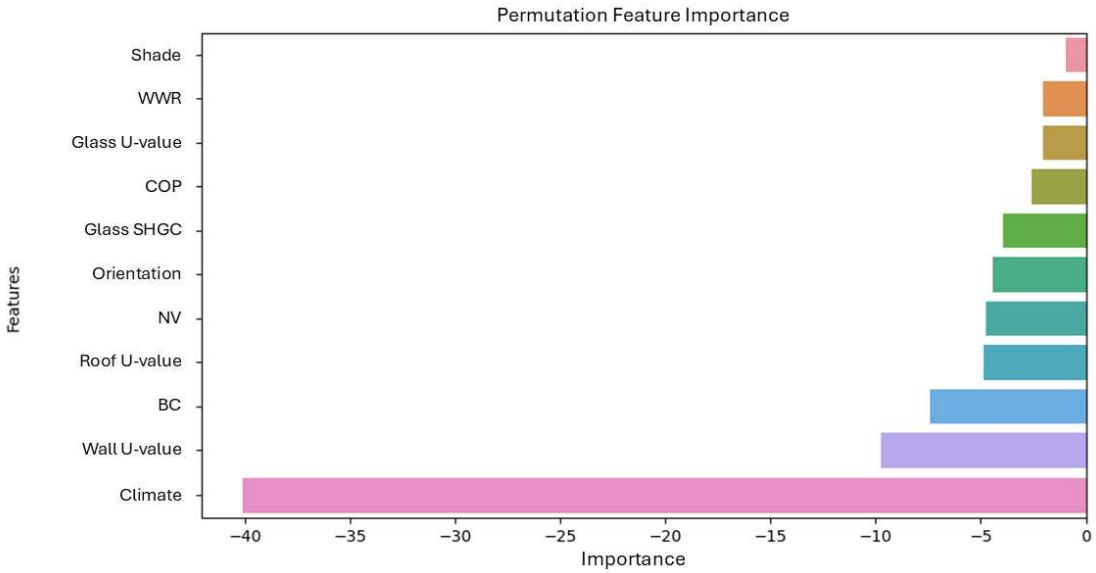


Figure 7. Permutation feature importance chart for ANN model.

Intermediate values for the top three important numeric features i.e. Wall Type, Roof Type and Orientation of ANN were selected. In addition, we also simulated AC Coefficient of Performance. Table 4 shows the intermediate values used to obtain the unseen validation dataset.

Table 4. Intermediate values are modelled to obtain unseen validation datasets.

Parameter	Intermediate Value
Wall Type (U-value: W/sqmK)	1.112
Roof Type (U-value: W/sqmK)	0.632
Orientation (degree)	45
AC (Coefficient of Performance)	2.7

To obtain a validation dataset for comparison, these intermediate values were fed into a simulation model with various configurations of other parameters. We approached the process methodically, first modeling 1 unseen wall type along with 2 roof types, 2 orientations, and 2 AC performance levels from original dataset. Other variables such as glazing types, Window-to-Wall Ratio (WWR), shading elements, and climate zones remained consistent with the original setup. This resulted in a total of 1,080 parametric runs.

Next, we adjusted the simulation to include 1 unseen roof type, while keeping the same known (02 Nos.) wall, orientations and AC performance levels, which also produced 1,080 runs. We repeated this approach for the orientation and AC performance cases, which brought the total number of parametric runs to 4,320 (1,080 runs per case, multiplied by 4 variations-Table 5). Additionally, we limited categorical parameters like ventilation strategies and building configurations to only one type each. These simulation run time for these cases (Figure 8) took ~ 5-6 days. This approach of restricting parameters such as wall type, roof type, orientation, and AC performance to only two values from the original dataset ensured that the simulations were both manageable and still sufficiently representative.

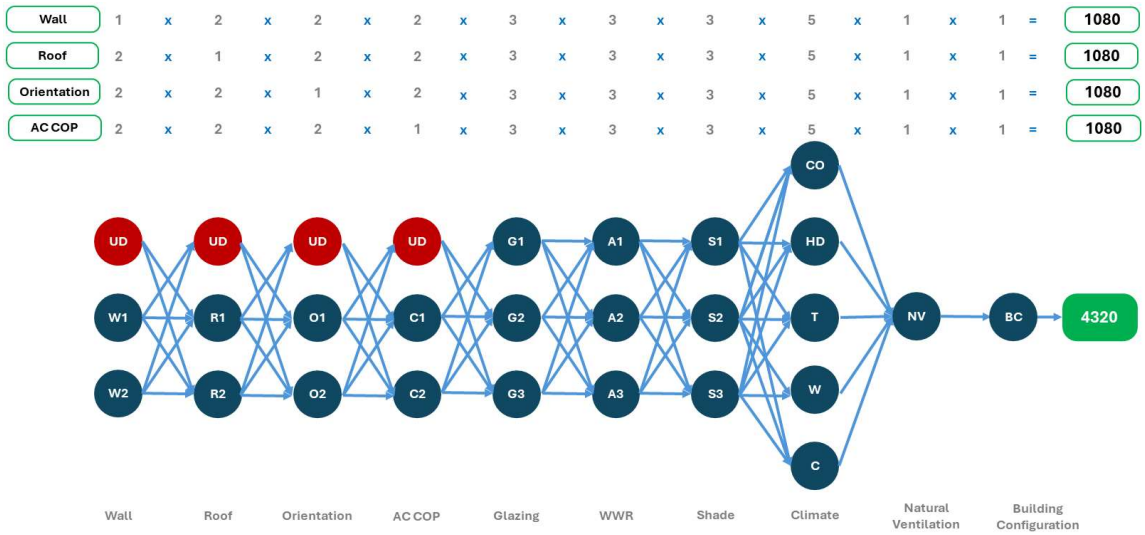


Figure 8. Number of parametric runs conducted with unseen dataset (UD: Unseen Datavalue).

The analysis of the percentage differences between the actual and predicted values for the unseen dataset, which was not part of the training dataset, revealed important insights into the model's performance. With an average percentage difference of -1.2%, the model's predictions tended to slightly underestimate the actual values, indicating a consistent bias in the predictions towards the lower end. This slight negative average suggested that, while the model was generally close to the actual values, there could be systematic factors leading to these underestimations. Table 6 provides a comparison between the Simulated EPI values and the Predicted EPI values for both the unseen and the original datasets.

Table 5. Number of runs and input values of parametric runs.

Parameter	Wall	Roof	Orientation	AC COP
Wall Type (U-value: W/sqmK)	1 (1.112)	2 (2.256 & 0.719)	2 (2.256 & 0.719)	2 (2.256 & 0.719)
Roof Type (U-value: W/sqmK)	2 (0.869 & 0.492)	1 (0.632)	2 (0.869 & 0.492)	2 (0.869 & 0.492)
the Orientation (degree)	2 (0 & 90)	2 (0 & 90)	1 (45)	2 (0 & 90)
AC (Coefficient of Performance)	2 (2.4 & 3.0)	2 (2.4 & 3.0)	2 (2.4 & 3.0)	1 (3.7)
Glass (U-value: W/sqmK) / SHGC	3 (Original Datapoints)	3 (Original Datapoints)	3 (Original Datapoints)	3 (Original Datapoints)
Window to Wall Ratio	3 (Original Datapoints)	3 (Original Datapoints)	3 (Original Datapoints)	3 (Original Datapoints)
Shade	3 (Original Datapoints)	3 (Original Datapoints)	3 (Original Datapoints)	3 (Original Datapoints)
Natural Ventilation	1 (Yes)	1 (Yes)	1 (Yes)	1 (Yes)
Building Configuration	1 (Stand Alone)	1 (Stand Alone)	1 (Stand Alone)	1 (Stand Alone)
Climate	5 (Original Dataset)	5 (Original Dataset)	5 (Original Dataset)	5 (Original Dataset)
Total Runs	1080	1080	1080	1080

Table 6. Simulated EPI vs. Predicted EPI for selected values of unseen and original datasets.

S.No.	Climate	Wall U-value	Roof U-value	Glass U-value	Glass SHGC	WWR	Shade	Orientation	COP	NV	BC	Simulated EPI	Predicted EPI
0	Composite	2.256	0.869	5.591	0.58	0.3	0.5	0	2.4	No	Stand-Alone	118.500	118.799
1	Composite	1.112	0.869	5.591	0.58	0.3	0.5	0	2.4	No	Stand-Alone	96.791	99.630
2	Composite	0.719	0.869	5.591	0.58	0.3	0.5	0	2.4	No	Stand-Alone	93.426	91.949
3	Composite	2.256	2.865	5.591	0.58	0.3	0.5	0	2.4	No	Stand-Alone	122.850	121.986
4	Composite	2.256	0.632	5.591	0.58	0.3	0.5	0	2.4	No	Stand-Alone	116.634	115.093
5	Composite	2.256	0.492	5.591	0.58	0.3	0.5	0	2.4	No	Stand-Alone	106.929	106.808
6	Composite	2.256	0.869	5.591	0.58	0.3	0.5	45	2.4	No	Stand-Alone	120.487	121.317
7	Composite	2.256	0.869	5.591	0.58	0.3	0.5	90	2.4	No	Stand-Alone	126.545	126.817
8	Composite	2.256	0.869	5.591	0.58	0.3	0.5	180	2.4	No	Stand-Alone	119.652	119.076
9	Composite	2.256	0.869	5.591	0.58	0.3	0.5	0	2.7	No	Stand-Alone	116.483	115.371
10	Composite	2.256	0.869	5.591	0.58	0.3	0.5	0	3	No	Stand-Alone	114.823	114.362

The standard deviation of 2.49% (Table 7) indicated a relatively low variability in the percentage differences, suggesting that most predictions were closely clustered around the average. This low standard deviation reflected consistent performance across the dataset, demonstrating that the model's predictions were generally reliable and exhibited minimal fluctuations from the average prediction error.

Table 7. Average % Difference and Standard Deviation.

Parameter	Value
Average % Difference	-1.2%
Standard Deviation	2.49%

However, some discrepancies could arise from factors such as model limitations or external influences not accounted for in the analysis. While the model demonstrated good reliability, there are opportunities for refinement to enhance predictive accuracy. Increasing the original dataset could significantly improve the model's ability to capture underlying patterns and reduce prediction errors. By expanding the dataset to include a broader range of examples and scenarios, more accurate and dependable predictions in future analyses may be achievable, thereby increasing the model's overall performance and robustness.

5 Conclusion

In this study, we developed a set of surrogate ML models to investigate their potential in predicting the energy performance index (EPI) for a standalone residential prototype. We trained and evaluated seven standard ML models and an ANN model. Through comprehensive experimentation and analysis, we determined that the ANN-based model yielded the best MAE and MAPE performance metrics of 0.702303 and 0.010877 respectively. The results of classical models indicated that non-linear models, especially the XGBoost, Random Forest Regressor and Decision Tree model, significantly outperformed linear models. This outcome strongly suggests the superiority of ANNs in accurately predicting residential thermal behavior compared to the other classical ML models. These research findings indicate that leveraging ML techniques holds promise for early design stages of residential building design. This approach provides insights into which specific building features most significantly impact the building's Energy Performance Index. These models will not only help designers but also prove valuable for policymakers by helping them understand the carbon saving potential of different strategies in designing residential buildings. It is worth noting that further investigations can be conducted to explore the potential of the other

ML models considered in this study. Additionally, future research endeavors may involve fine-tuning the ANN model and exploring advanced techniques to enhance its performance and applicability to real-world residential thermal modeling scenarios.

In conclusion, our study underscores the efficacy of ML, particularly the TensorFlow-based ANN model as a valuable tool for residential thermal modeling. The findings highlight the significance of leveraging ML techniques to enhance our understanding of residential thermal behavior and inform more efficient, sustainable building design and energy management strategies.

6 Acknowledgments

We would like to thank the Environmental Design Solutions team for providing the dataset for training and testing the surrogate model for residential buildings.

7 Data availability statement

The data that support the findings of this study are available from the corresponding author, MK, upon reasonable request.

References

1. Ramy Mahmoud, John M. Kamara, Neil Burford. Opportunities and limitations of building energy performance simulation tools in the early stages of building design in the UK. Sustainability, 12(22):2020. https://doi.org/10.3390/su12229702

2. Zhihong Pang, Zheng O'Neill, Yanfei Li, Fuxin Niu. The role of sensitivity analysis in the building performance analysis: A critical review. Energy and Buildings, 209:109659, 2020. https://doi.org/10.1016/j.enbuild.2019.109659

3. Mariana Migliori, Hamidreza Najafi, Aldo Fabregas, Troy Nguyen. Neural network-based building energy models for adapting to post-occupancy conditions: A case study for florida. Journal of Engineering for Sustainable Buildings and Cities, 3(4):041002, 2022. https://doi.org/10.1115/1.4056393

4. Kangji Li, Lei Pan, Wenping Xue, Hui Jiang, Hanping Mao. Multi objective optimization for energy performance improvement of residential buildings: A comparative study. Energies, 10(2):2017. https://doi.org/10.3390/en10020245

5. Laurent Magnier, Fariborz Haghighat. Multi objective optimization of building design using trnsys simulations, genetic algorithm, and artificial neural network. Building and Environment, 45(3):739–746, 2010. https://doi.org/10.1016/j.buildenv.2009.08.016

6. Didier Gossard, Berangere Lartigue, Francoise Thellier. Multi-objective optimization of a building envelope for thermal performance using genetic algorithms and artificial neural network. Energy and Buildings, 67:253–260, 2013. https://doi.org/10.1016/j.enbuild.2013.08.026

7. Ehsan Asadi, Manuel Gameiro da Silva, Carlos Henggeler Antunes, Luis Dias. Multi-objective optimization for building retrofit strategies: A model and an application. Energy and Buildings, 44:81–87, 2012. https://doi.org/10.1016/j.enbuild.2011.10.016

8. Hui Zou, Trevor Hastie. Regularization and Variable Selection Via the ElasticNet. Journal of the Royal Statistical Society Series B: Statistical Methodology, 67(2):301–320, 03 2005. https://doi.org/10.1111/j.1467-9868.2005.00503.x.

9. Quentin Klopfenstein, Samuel Vaiter. Linear support vector regression with linear constraints. Machine Learning, 110(7):1939–1974, Jul 2021. https://doi.org/10.1007/s10994-021-06018-2

10. Leo Breiman. Random forests. Machine Learning, 45(1):5–32, Oct 2001. https://doi.org/10.1023/A:1010933404324

11. Tianqi Chen, Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery. https://doi.org/10.1145/2939672.2939785