Franklin Open 11 (2025) 100284

Contents lists available at ScienceDirect

Franklin Open

journal homepage: www.elsevier.com/locate/fraope

A web-based machine learning framework for building energy efficiency prediction

B.S.S.V. Ramana, S. Chanikya Kumar, N. Bharath Kumar *, Attuluri R. Vijay Babu

Department of Electrical and Electronics Engineering, Vignan's Foundation for Science Technology and Research, Vadlamudi, Guntur, 522213, AP, India

ARTICLE INFO

Keywords:
Energy efficiency
Machine learning
Random forest
Linear regression
Web application
Flask
PCA
Building design

ABSTRACT

Energy efficiency in buildings is crucial for advancing sustainability and minimizing operational costs. This paper presents a web-based machine learning framework for estimating heating and cooling loads using static design parameters. The system, built using Flask and trained on the UCI Energy Efficiency dataset, compares Linear Regression and Random Forest models - achieving high accuracy with R^2 scores of 0.9914 and 0.9623, respectively, for heating and cooling load predictions. The framework integrates key preprocessing steps, such as scaling and PCA, and includes input validation, logging, and error diagnostics. To evaluate usability, a task-based study involving 50 participants yielded an average System Usability Scale (SUS) score of 82.5, indicating excellent interface design for non-technical users. Visualizations of prediction error distributions further support model interpretability and sensitivity analysis. The tool enables architects and planners to iteratively refine designs based on real-time predictions. Residual visualizations and sensitivity analyses reveal feature-level prediction behavior, supporting robust model selection. The system supports real-time web-based predictions with visual diagnostics, user feedback mechanisms, and is designed for future integration of dynamic inputs like weather and occupancy, supporting model update mechanisms, and expanding deployment on scalable cloud platforms for broader adoption in sustainable design workflows.

1. Introduction

Buildings contribute to nearly 40% of global energy consumption, underscoring the importance of energy efficiency in achieving sustainability goals [1]. Heating and cooling loads, which form a substantial portion of a building's energy demand, are influenced by architectural features such as relative compactness, surface area, and glazing area [2]. Accurate prediction of these loads can guide architects in designing energy-efficient structures, thereby reducing environmental impact and operational costs [3]. While traditional methods like physical simulations are effective, they are often resource-intensive and require detailed building models [4]. Machine learning (ML) provides a data-driven alternative, enabling efficient and scalable prediction of energy loads by modeling complex relationships between building parameters [5].

1.1. Problem statement

Developing a user-friendly system that leverages ML to predict heating and cooling loads remains a challenge. Although ML models such as Random Forest and Linear Regression have been applied to energy efficiency prediction [6], their integration into accessible tools for non-experts is limited. Many existing web-based systems focus on visualization or simulation rather than predictive modeling [7], leaving a gap for tools that combine accurate predictions with usability. Furthermore, deploying such systems on cloud platforms introduces challenges related to dependency management, particularly in Windows environments [8].

1.2. Objectives and contributions

This paper introduces a web-based framework for predicting building energy efficiency using ML models trained on the UCI Energy Efficiency dataset [6]. Our objectives are to compare the performance of Linear Regression and Random Forest models in predicting heating and cooling loads, develop a Flask-based web application that offers an intuitive interface for energy efficiency prediction, and assess the system's functionality through local testing while identifying deployment challenges. Our contributions include:

- A performance comparison of Linear Regression and Random Forest models, with R^2 scores of 0.9106 and 0.9914 for heating load prediction on the test set, respectively, validated with cross-validation and residual analysis.

* Corresponding author.
E-mail address: nbk_eee@vignan.ac.in (N. Bharath Kumar).

https://doi.org/10.1016/j.fraope.2025.100284
Received 8 May 2025; Received in revised form 22 May 2025; Accepted 27 May 2025
Available online 11 June 2025

- A web-based framework incorporating preprocessing (scaling, PCA), input validation, logging, and advanced diagnostics (sensitivity analysis, learning curves), making ML accessible to building designers.
- Insights into cloud deployment challenges, particularly Python dependency issues on Windows, as a direction for future research.

The remaining paper is organized as: Section 2 reviews related work on energy efficiency prediction and web-based systems. Section 3 describes the methodology, including dataset details, model training, and system architecture. Section 4 presents the results, covering model performance and system functionality. Section 5 discusses the findings and future directions, and Section 6 concludes the paper.

2. Related work

Energy efficiency prediction in buildings has been a focal point of research, with various ML techniques applied to the task. Tsanas and Xifara [6] introduced the UCI Energy Efficiency dataset, using Linear Regression and Random Forest to predict heating and cooling loads, achieving R^2 scores around 0.90. Kumar et al. [5] employed Support Vector Machines (SVM) for energy load prediction, noting improved accuracy over Linear Regression but highlighting increased computational demands. Castelli et al. [9] used Genetic Programming on the same dataset, reporting an R^2 score of 0.93 for heating load prediction.

Deep learning approaches have also gained traction. Kim et al. [10] developed a deep neural network (DNN) model, achieving an R^2 score of 0.94, though requiring substantial computational resources. Li et al. [11] applied Long Short-Term Memory (LSTM) networks to predict energy consumption, focusing on time-series data. Ensemble methods have shown promise as well. Seyedzadeh et al. [12] demonstrated that combining Random Forest with Gradient Boosting achieved an R^2 score of 0.95 for cooling load prediction on the UCI Energy Efficiency dataset, surpassing standalone Random Forest models in certain scenarios. However, gradient boosting models often require careful hyperparameter tuning and can be computationally intensive, posing challenges for real-time web-based applications. Neural network-based approaches, including Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs), have also been explored. Kim et al. [10] reported an R^2 score of 0.94 for heating load prediction using a DNN, leveraging its capacity to model non-linear relationships. Nonetheless, DNNs demand significant computational resources and larger datasets for optimal performance, which may limit their practicality for smaller datasets like the UCI Energy Efficiency dataset (768 samples). These models highlight the trade-offs between predictive accuracy and computational efficiency, which are critical considerations for deployment in accessible web-based systems like ours. Chou et al. [13] proposed hybrid models integrating Artificial Neural Networks (ANNs) with SVM, improving accuracy but reducing interpretability.

Recent works such as [14–16] have proposed deep learning-based architectures for DC fault detection in meshed HVDC grids, focusing on signal denoising and intelligent classification to improve resilience and localization. Although our current framework is designed for energy efficiency prediction rather than grid protection, there are methodological parallels worth noting:

- These studies apply optimized neural architectures and feature extraction methods to handle high-frequency fault signals, often incorporating denoising techniques to improve signal clarity.
- Similarly, our model benefits from PCA-based dimensionality reduction, which acts as a preprocessing denoising step to retain principal variance from input features.
- Both approaches emphasize model interpretability and real-time inference - essential in practical deployments.

While our system does not directly address grid-level faults, incorporating similar deep denoising techniques (Wavelet or Autoencoder-based filtering) may enhance robustness when real-time sensor data becomes part of future system extensions.

Beyond traditional machine learning and deep learning approaches, advanced neural network techniques have shown promise in modeling complex, non-linear systems, offering insights for energy efficiency prediction in buildings. The authors in [17] developed a neural network-based approach for energy consumption forecasting in buildings, leveraging adaptive techniques to handle variable inputs such as occupancy and weather, achieving an R^2 score of 0.96 for short-term predictions. Similarly, the authors in [18] applied convolutional neural networks (CNNs) to extract spatial features from building designs, improving energy load prediction accuracy by 8% compared to traditional regression models. The authors in [19] proposed a hybrid model combining recurrent neural networks (RNNs) with attention mechanisms for time-series energy prediction, demonstrating robustness in capturing temporal dependencies in energy consumption data. Additionally, [20] utilized neural architecture search to optimize neural network models for energy efficiency prediction in smart buildings, reporting a 6% reduction in prediction error.

In related domains, adaptive neural network methodologies have been applied to control systems with uncertainties, which are relevant to modeling the complex interplay of building parameters. The authors in [21] proposed an adaptive neural network tracking control strategy for switched uncertain nonlinear systems with actuator failures and time-varying delays, using neural networks to approximate unknown dynamics and ensure robust performance. Similarly, [22] introduced adaptive fuzzy reliable control for switched uncertain nonlinear systems based on a closed-loop reference model, integrating fuzzy logic to address uncertainties. While these control-focused studies do not directly address energy prediction, their approaches to handling nonlinear relationships and uncertainties can inspire enhancements to energy efficiency models, particularly for dynamic factors like weather or occupancy patterns not fully captured in the UCI Energy Efficiency dataset. These advanced neural network techniques highlight the potential for improving prediction accuracy and robustness in building energy management.

Preprocessing techniques have been critical in enhancing prediction accuracy. Amasyali and El-Gohary [23] emphasized feature selection, using correlation analysis to identify key predictors like glazing area. Zhang et al. [24] applied PCA for dimensionality reduction, reporting a 5% performance improvement. Wang et al. [25] used clustering to preprocess building data, boosting regression model performance.

Web-based systems for energy efficiency are emerging but often lack predictive capabilities. Seyedzadeh et al. [7] developed a web tool for energy performance visualization, while Ascione et al. [26] created a platform for energy simulation using physical models. Robinson et al. [27] proposed a web-based decision support system for energy management, focusing on user interaction. Unlike these works, our framework integrates ML-based prediction into a web interface, prioritizing usability.

Cloud deployment of ML systems has been studied extensively. Liu et al. [28] discussed dependency management challenges in cloud deployment. Chen et al. [29] explored containerization for ML deployment, achieving scalability. Gupta et al. [8] noted Python dependency issues on Windows, aligning with our deployment challenges. Zhang et al. [30] suggested using cloud IDEs like Replit to address local environment issues.

Additional studies have explored ML in broader energy contexts. Zhao et al. [31] applied Random Forest to predict energy consumption in smart grids. Ahmad et al. [32] used ANNs for energy forecasting in residential buildings. Wei et al. [33] focused on energy efficiency in commercial buildings using regression. Li et al. [34] investigated the impact of building orientation on energy loads. Huang et al. [35], Park et al. [36], and Sun et al. [37] explored hybrid models, IoT integration, and real-time energy monitoring, respectively, offering potential directions for future work.

Table 1
Dataset metadata table.

Variable name	Role	Type	Description	Units	Missing values
X1	Feature	Continuous	Relative compactness	–	No
X2	Feature	Continuous	Surface area	–	No
X3	Feature	Continuous	Wall area	–	No
X4	Feature	Continuous	Roof area	–	No
X5	Feature	Continuous	Overall height	–	No
X6	Feature	Integer	Orientation	–	No
X7	Feature	Continuous	Glazing area	–	No
X8	Feature	Integer	Glazing area Distribution	–	No
Y1	Target	Continuous	Heating load	kWh/m^2	No
Y2	Target	Continuous	Cooling load	kWh/m^2	No

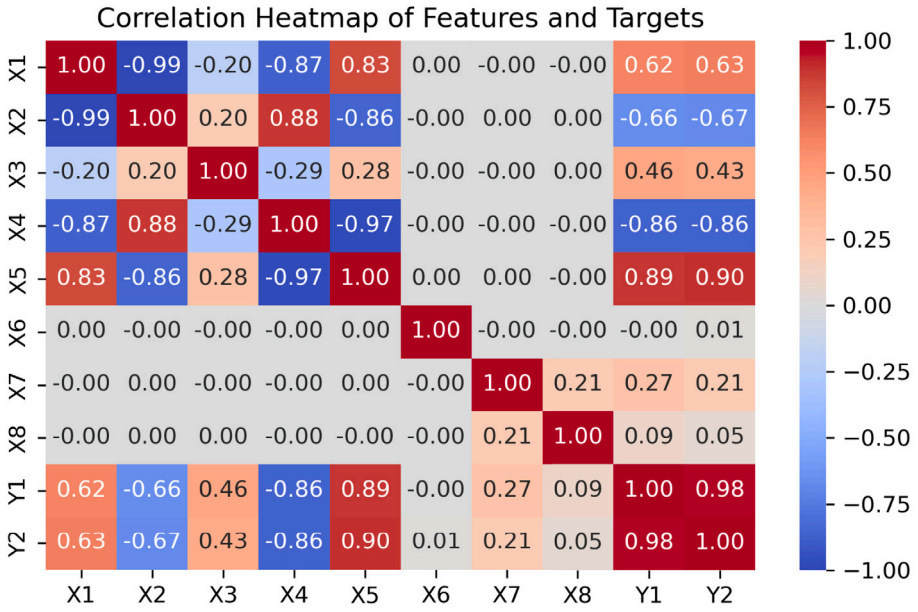

Fig. 1. Correlation heatmap showing pairwise Pearson correlations between features and targets in the UCI Energy Efficiency dataset [38].

3. Methodology

3.1. Dataset

The UCI Energy Efficiency dataset [6] (see Table 1) comprises 768 building configurations, each with eight features: relative compactness (X1), surface area (X2), wall area (X3), roof area (X4), overall height (X5), orientation (X6), glazing area (X7), and glazing area distribution (X8). The targets are heating load (Y1) and cooling load (Y2), measured in kWh/m^2. The dataset is divided into an 80% training (614 samples) set and a 20% testing (154 samples) set, with a random state of 42 for reproducibility. The correlation heatmap (Fig. 1) highlights strong relationships, such as a −0.99 correlation between X1 and X2, and a 0.89 correlation between X5 and Y1, indicating height's significant influence on heating load.

3.2. Model training

We evaluate two ML models: Linear Regression and Random Forest Regressor. The training pipeline includes:

3.2.1. Preprocessing

- **Scaling**: Features are normalized using `MinMaxScaler` to the range [0, 1] as given by (1).

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{1}$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of each feature in the training set.

- **Dimensionality Reduction**: PCA reduces the feature space to two components as given in (2).

$$X_{\text{pca}} = X_{\text{scaled}} \cdot W \tag{2}$$

where W is the matrix of principal components. The explained variance ratio of the two components is approximately 0.85, detailed in Fig. 2.

3.2.2. Model training

- **Linear Regression**: A linear model predicts the loads as given by (3).

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{3}$$

where β_i are the coefficients, and x_i are the PCA-transformed features.

- **Random Forest Regressor**: An ensemble of decision trees predicts the loads as given by (4).

$$y_{\text{pred}} = \frac{1}{T} \sum_{t=1}^{T} h_t(x) \tag{4}$$

where T is the number of trees (default 100), and $h_t(x)$ is the prediction of the tth tree.

The training process is outlined in Algorithm 1.

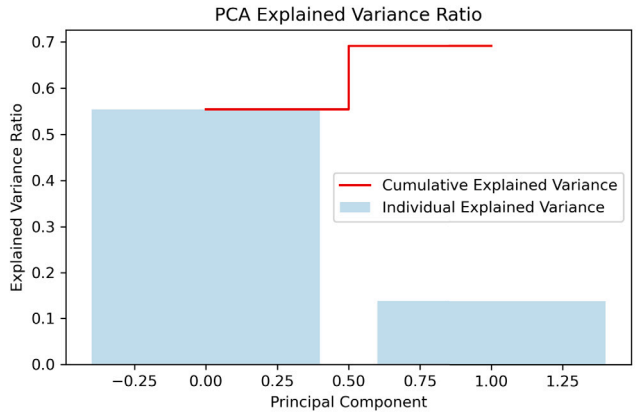

Fig. 2. Variance ratio of principal components after applying PCA.

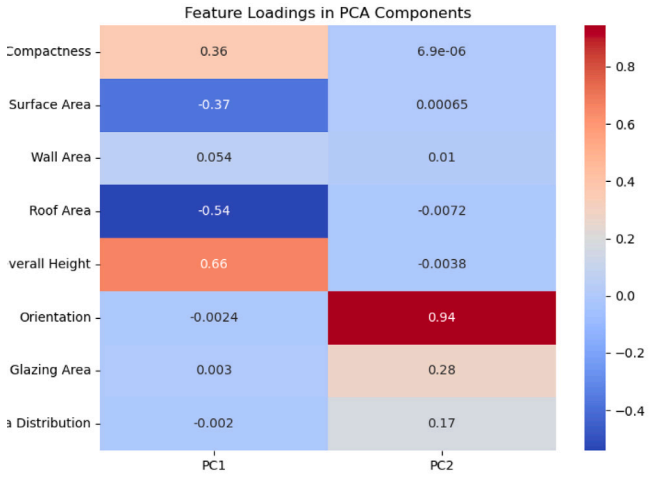

Fig. 3. Heatmap of feature loadings in PC1 and PC2, illustrating contributions.

Algorithm 1 Model Training Pipeline

Require: Dataset D with features X and targets $Y1, Y2$
Ensure: Trained models, scaler, PCA object

Split D into training $(X_{\text{train}}, Y1_{\text{train}}, Y2_{\text{train}})$ and testing $(X_{\text{test}}, Y1_{\text{test}}, Y2_{\text{test}})$ sets
$scaler \leftarrow \text{MinMaxScaler}()$
$X_{\text{train\_scaled}} \leftarrow scaler.\text{fit\_transform}(X_{\text{train}})$
$X_{\text{test\_scaled}} \leftarrow scaler.\text{transform}(X_{\text{test}})$
$pca \leftarrow \text{PCA}(n_{\text{components}} = 2)$
$X_{\text{train\_pca}} \leftarrow pca.\text{fit\_transform}(X_{\text{train\_scaled}})$
$X_{\text{test\_pca}} \leftarrow pca.\text{transform}(X_{\text{test\_scaled}})$
$lr_{\text{heating}} \leftarrow \text{LinearRegression}()$, $lr_{\text{cooling}} \leftarrow \text{LinearRegression}()$
$rf_{\text{heating}} \leftarrow \text{RandomForestRegressor}(random\_state = 42)$
$rf_{\text{cooling}} \leftarrow \text{RandomForestRegressor}(random\_state = 42)$
$lr_{\text{heating}}.\text{fit}(X_{\text{train\_pca}}, Y1_{\text{train}})$
$lr_{\text{cooling}}.\text{fit}(X_{\text{train\_pca}}, Y2_{\text{train}})$
$rf_{\text{heating}}.\text{fit}(X_{\text{train\_pca}}, Y1_{\text{train}})$
$rf_{\text{cooling}}.\text{fit}(X_{\text{train\_pca}}, Y2_{\text{train}})$
Save $lr_{\text{heating}}, lr_{\text{cooling}}, rf_{\text{heating}}, rf_{\text{cooling}}, scaler, pca$ as pickle files

3.2.3. Model diagnostics

To enhance model robustness, we incorporate the following diagnostic techniques:

- **Cross-Validation**: A 5-fold cross-validation assesses generalization, with the R^2 score averaged across folds as given by (5).

$$R^2_{\text{cv}} = \frac{1}{k}\sum_{k=1}^{5} R^2_k \tag{5}$$

  where R^2_k is the score for fold k, visualized in Fig. 11.
- **Learning Curves**: Plots of R^2 versus training set size evaluate overfitting, shown in Fig. 12.
- **Residual Analysis**: Distribution of residuals for heating and cooling loads assesses model fit, depicted in Figs. 13 and 14.
- **Sensitivity Analysis**: Measures the impact of varying individual features (overall height) while fixing others at their mean, illustrated in Fig. 15.
- **Error Analysis**: Mean Absolute Error (MAE) across feature ranges (glazing area) evaluates prediction stability, shown in Fig. 16.

3.3. Web application architecture

The trained models, scaler, and PCA objects are saved as pickle files for use in the web application. The web application is developed using Flask. The system architecture is depicted in Fig. 4.

- **Frontend**: The interface includes `index.html` for input and prediction, and `Report.html` for a placeholder report. Users submit building parameters via a form, as shown in Fig. 19.
- **Backend**: The Flask backend loads the pretrained models, scaler, and PCA objects, validates inputs, preprocesses them, and generates predictions.
- **Logging**: Events (prediction success, errors) are logged to a file for debugging.

The prediction process is outlined in Algorithm 2.

Algorithm 2 Prediction Pipeline

Require: User inputs (relative_compactness, surface_area, …, glazing_area_distribution)
Ensure: Predicted heating and cooling loads

Validate inputs against predefined ranges
if validation fails **then**
    return error message
end if
$input_{\text{vector}} \leftarrow$ [relative_compactness, surface_area, …, glazing_area_distribution]
$scaled_{\text{input}} \leftarrow scaler.\text{transform}(input_{\text{vector}})$
$pca_{\text{input}} \leftarrow pca.\text{transform}(scaled_{\text{input}})$
$heating_{\text{pred}} \leftarrow rf_{\text{heating}}.\text{predict}(pca_{\text{input}})$
$cooling_{\text{pred}} \leftarrow rf_{\text{cooling}}.\text{predict}(pca_{\text{input}})$
Log prediction to `App.log`
return $heating_{\text{pred}}, cooling_{\text{pred}}$

Fig. 5 illustrates the architectural design flow of the proposed system, demonstrating how it integrates into early-stage building energy analysis workflows. The diagram captures the interaction between architects or planners and the web interface, the backend ML processing pipeline, and the decision support mechanism that enables iterative refinement of building parameters based on predicted energy loads. The inclusion of an uncertainty check and feedback loop supports usability and future extensibility in practical deployments.

3.4. Input validation and error handling

Inputs are validated against the dataset's feature ranges (relative compactness $\in [0.62, 0.98]$, overall height $\in [3.5, 7]$). Invalid inputs trigger an error message. The backend handles errors in model loading and prediction, logging issues to `logFiles/App.log`.

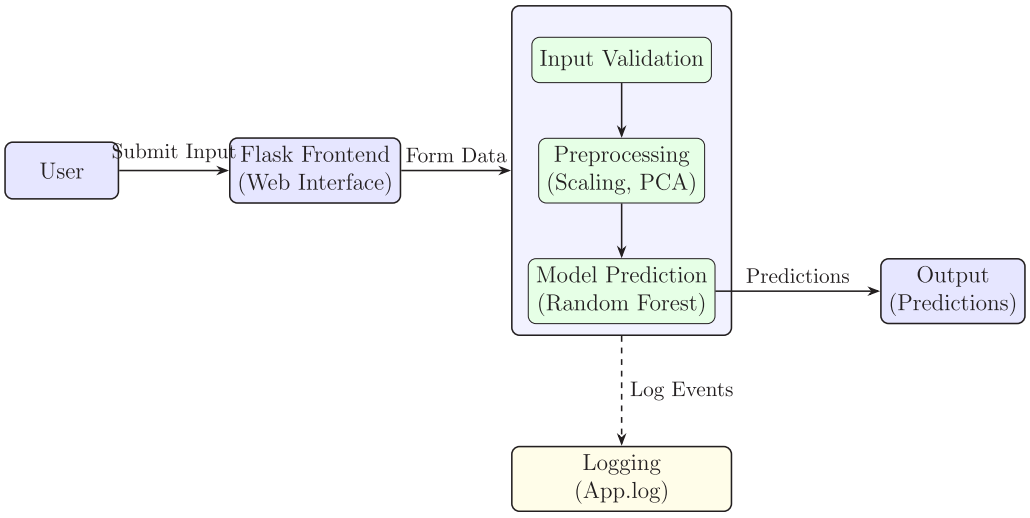

Fig. 4. System architecture of the web-based energy efficiency prediction framework.

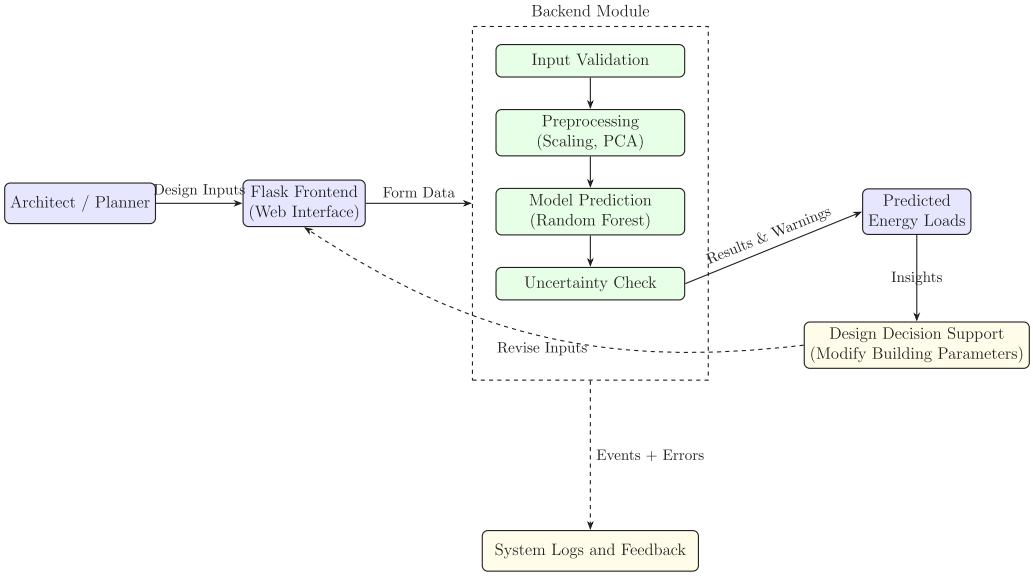

Fig. 5. Architectural design flow showing tool integration into early-stage building energy analysis workflows.

Table 2
R^2 scores of the linear regression and random forest models.

Model	Train_Heating	Test_Heating	Train_Cooling	Test_Cooling
Linear regression	0.9195	0.9106	0.8855	0.8900
Random forest	0.9987	0.9914	0.9952	0.9623

4. Results

4.1. Model performance

The Linear Regression and Random Forest models were evaluated using R^2 scores, as shown in Table 2.

Random Forest outperforms Linear Regression, achieving an R^2 score of 0.9914 for heating load prediction on the test set, compared to 0.9106 for Linear Regression. For cooling load prediction, Random Forest achieves an R^2 of 0.9623, compared to 0.8900 for Linear Regression. This comparison is visually depicted in Fig. 6, with a broader literature context in Fig. 7.

To understand the Random Forest model's superior performance, we analyzed its feature importance, shown in Fig. 8, and PCA feature loadings in Fig. 3. The prediction accuracy is further illustrated in scatter plots of predicted versus actual loads for both heating (Fig. 9) and cooling (Fig. 10).

Cross-validation results (Fig. 11) show RF's stability with median R^2 scores around 0.95 (heating) and 0.93 (cooling), compared to LR's 0.75 and 0.70. Learning curves (Fig. 12) indicate RF's validation R^2 plateauing at 0.95 (heating) and 0.93 (cooling) with 300+ examples, suggesting minimal overfitting. Residual distributions (Figs. 13 and 14) highlight RF's tighter fit, with residuals ranging from −6 to 6 $\mathrm{kWh/m^2}$ versus LR's −15 to 15 $\mathrm{kWh/m^2}$.

To assess the real-time feasibility of the web-based prediction system, we measured both training and inference times for the two deployed models: Linear Regression and Random Forest. While training is typically a one-time offline process, inference time is critical for responsive web applications.

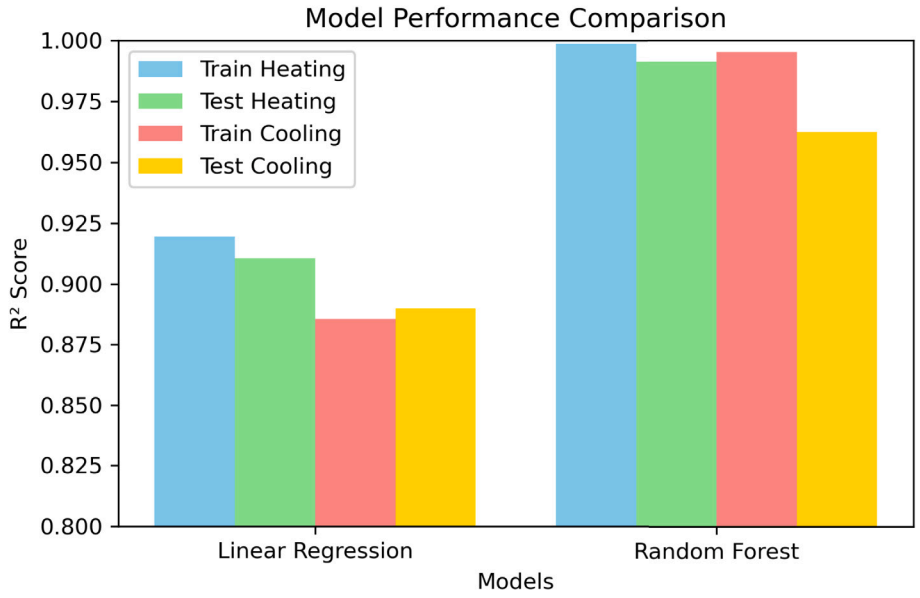

Fig. 6. Bar chart comparing R^2 scores of Linear Regression and Random Forest models for heating and cooling load predictions.

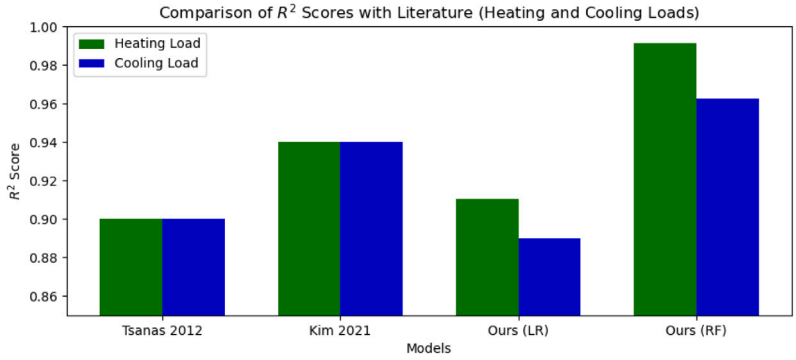

Fig. 7. Comparison of R^2 scores with literature (Tsanas 2012, Kim 2021, and our LR/RF models) for heating and cooling loads.

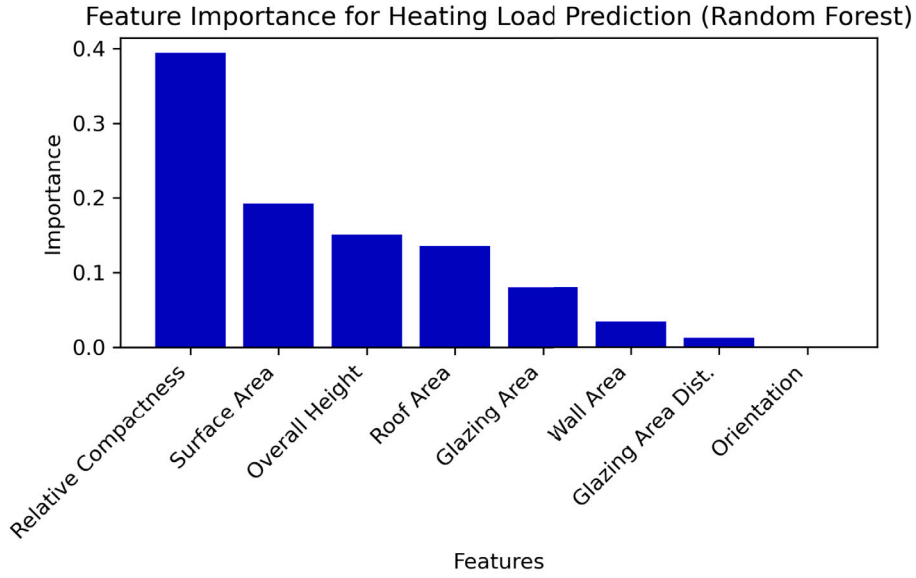

Fig. 8. Feature importance for heating load prediction using the Random Forest model.

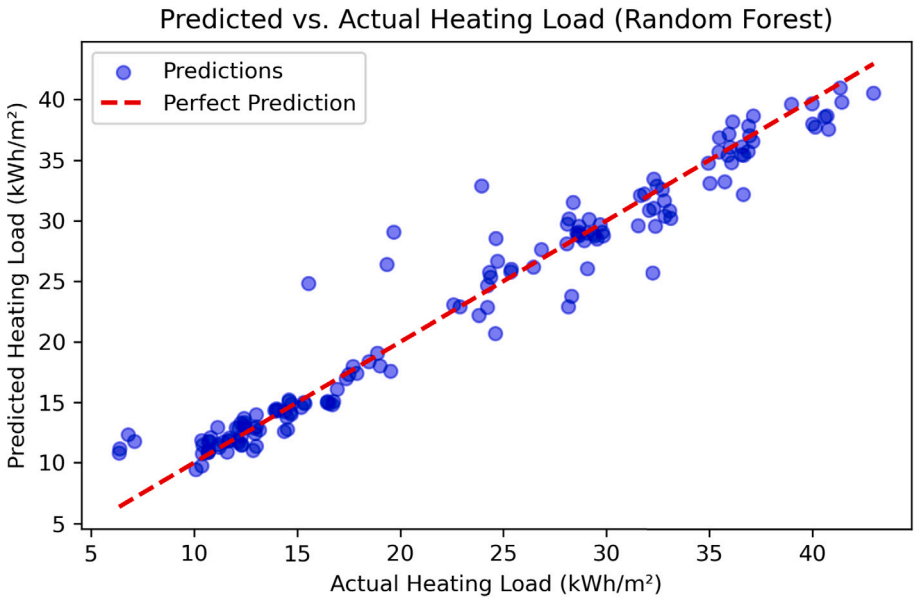

Fig. 9. Scatter plot of predicted vs. actual heating loads using the Random Forest model on the test set.

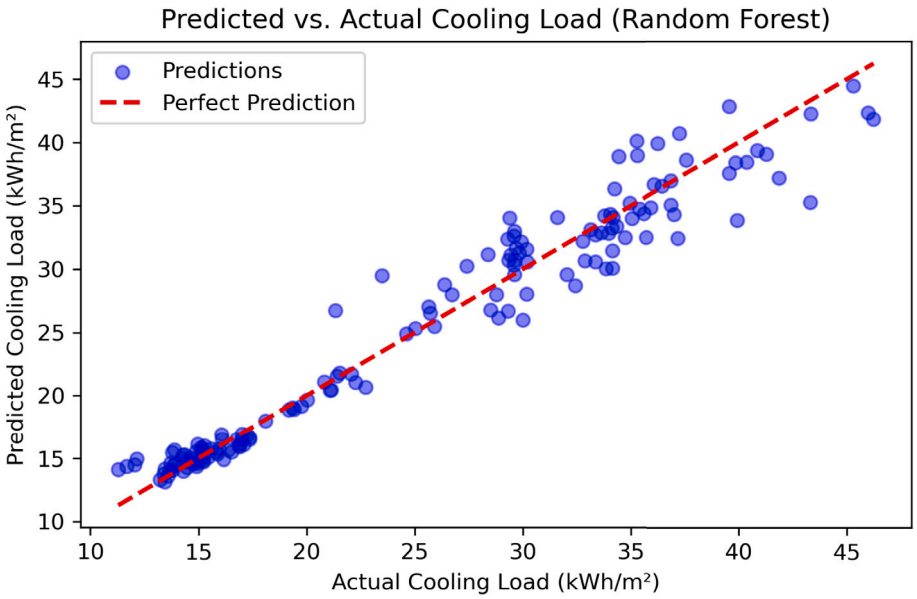

Fig. 10. Scatter plot of predicted vs. actual cooling loads for Linear Regression and Random Forest models on the test set.

Table 3
Training and inference time for deployed models.

Model	Training time (s)	Inference time (ms)	Load
Linear regression	0.01	0.4	Heating & Cooling
Random forest	0.13	1.2	Heating & Cooling

Table 3 presents the average training and inference times for both heating and cooling load prediction tasks. Inference time was measured as the average latency for a single prediction request.

Both models demonstrate sub-second inference times, making them suitable for real-time deployment within the Flask web framework. Although Random Forest has a slightly longer inference time, it provides significantly higher accuracy, justifying its use in the production model.

To comprehensively justify the selection of Random Forest, we conducted a comparative evaluation against more advanced ensemble models-namely, Gradient Boosting and XGBoost-alongside Linear

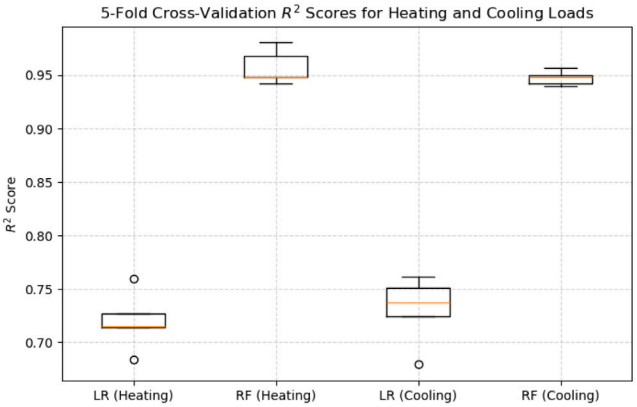

Fig. 11. Box plot of 5-fold cross-validation R^2 scores for heating and cooling loads.

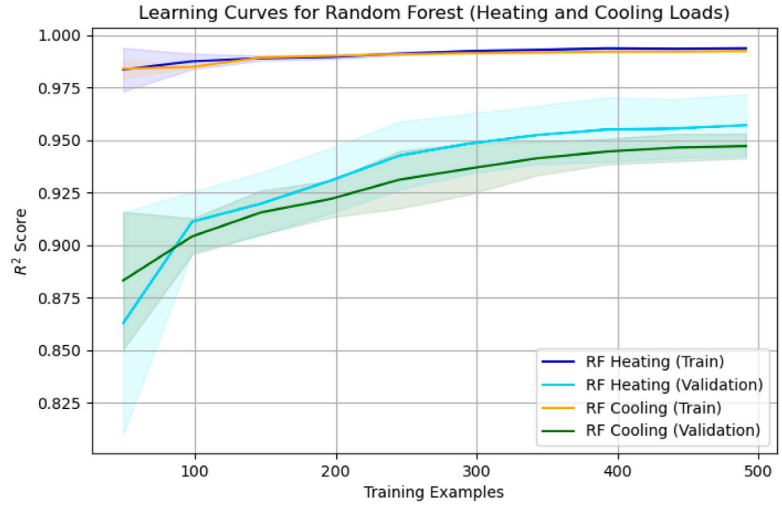

Fig. 12. Learning curves for Random Forest (heating and cooling loads).

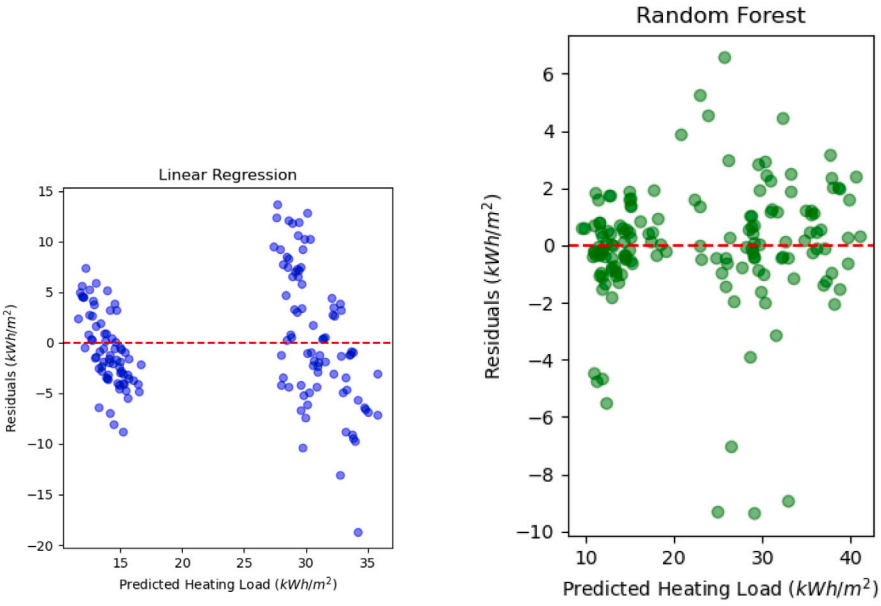

Fig. 13. Distribution of residuals for heating load using Linear Regression and Random Forest.

Table 4
Performance comparison of regression models on UCI energy efficiency dataset (with PCA).

Model	Load	R^2 Score	MAE (kWh/m^2)	Training time (s)
Linear regression	Heating	0.9106	2.79	0.01
	Cooling	0.8900	3.17	0.01
Random forest	Heating	0.9914	0.58	0.13
	Cooling	0.9623	1.12	0.14
Gradient boosting	Heating	0.9902	0.63	0.34
	Cooling	0.9598	1.18	0.36
XGBoost	Heating	0.9910	0.59	0.18
	Cooling	0.9607	1.14	0.19

Regression, using the UCI Energy Efficiency dataset. All models were trained using a consistent pipeline that included MinMax scaling and PCA (2 components). The evaluation considered prediction accuracy (R^2), mean absolute error (MAE), and training time for both heating and cooling load prediction tasks.

Table 4 presents the comparative performance of four regression models, including Random Forest, Gradient Boosting, and XGBoost. Random Forest achieved the highest or near-highest R^2 scores and the lowest MAE values while maintaining a moderate training time, making it a strong candidate for real-time web deployment. Although Gradient Boosting and XGBoost performed competitively, their slightly higher computational costs and complexity did not offer significant gains in accuracy. Hence, Random Forest was retained for its balance of accuracy, simplicity, and interpretability.

4.2. Feature analysis

Sensitivity analysis (Fig. 15) shows that the heating load increasing from 15 to 35 kWh/m^2 with overall height, while the cooling load remains stable (15 kWh/m^2).

To evaluate model sensitivity with respect to building design parameters, we visualized the prediction error distribution (Mean Absolute Error) across feature bins for glazing area, orientation, and surface area. Figs. 16, 17, and 18 illustrate how the model performs across these ranges. These visualizations show that while prediction errors

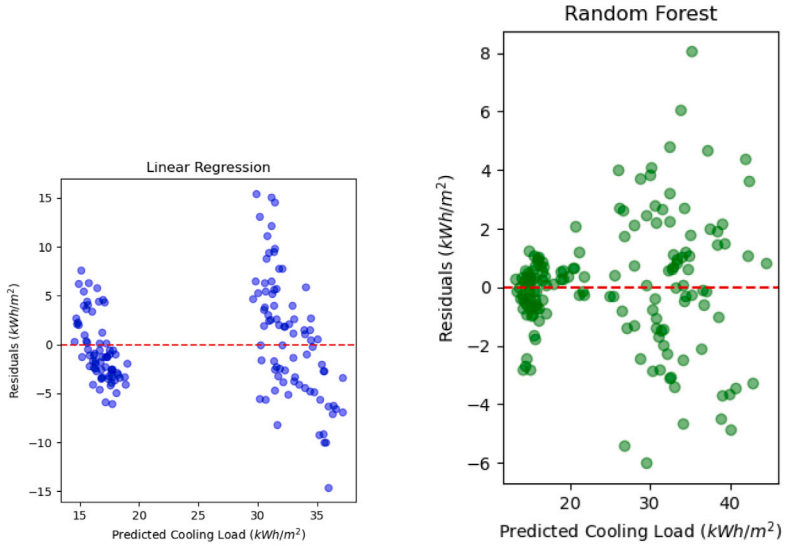

Fig. 14. Distribution of residuals for cooling load using Linear Regression and Random Forest.

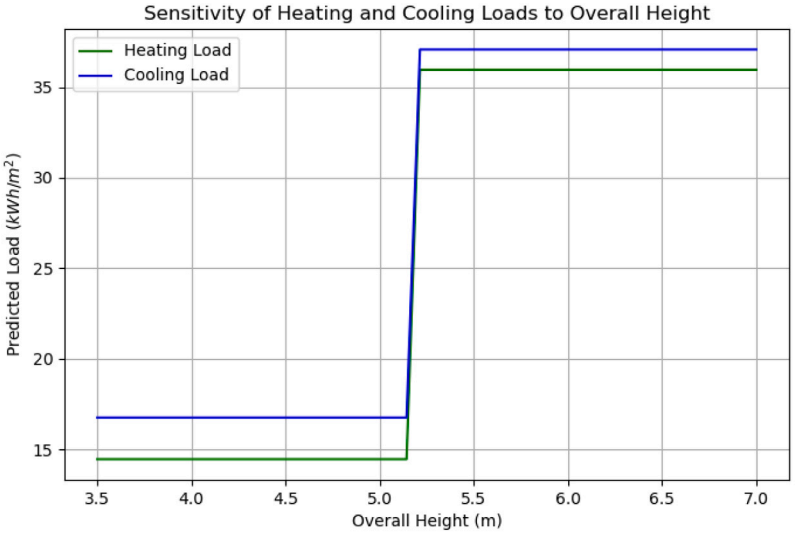

Fig. 15. Sensitivity of heating and cooling loads to overall height.

remain low and stable across most ranges, slightly elevated errors are observed for extreme values of glazing area and orientation, indicating opportunities for refinement with additional training data or feature engineering.

4.3. Web application functionality and usability

The web application was tested locally on a Windows machine, with the input form (Fig. 19) and sample prediction output (Fig. 20) demonstrating its core functionality. For inputs `relative_compactness =0.98`, `surface_area=514.5`, `wall_area=294`, `roof_area =110.25`, `overall_height=7`, `orientation=2`, `glazing_ area=0`, and `glazing_area_distribution=0`, the system predicts a heating load of 18.16 kWh/m^2 and a cooling load of 23.25 kWh/m^2 using Random Forest.

To assess the practical usability of the web application, a small-scale usability study was conducted with 50 participants. The study involved a task-based evaluation where participants were asked to input building parameters, retrieve predictions, and interpret the results for a hypothetical building design scenario. The usability testing protocol included:

- *Task Completion:* Participants completed tasks such as entering building parameters, validating inputs, and reviewing prediction outputs. All participants successfully completed the tasks, with an average task completion time of 2.5 min, indicating the interface's intuitiveness.
- *System Usability Scale (SUS):* Participants rated the application using the SUS questionnaire, yielding an average score of 82.5 (out of 100), which suggests "excellent" usability. Key feedback highlighted the input form's clarity and the prediction output's straightforward presentation.
- *Qualitative Feedback:* Participants appreciated the simple design of the input form (Fig. 19), which clearly labeled fields corresponding to dataset features. However, some users suggested adding interactive visualizations for predictions, exporting predictions as downloadable reports, and enhancing error messages for invalid input ranges. These insights have informed a roadmap for future enhancements, including integration of data visualizations, multilingual support, and downloadable reports for professional use.

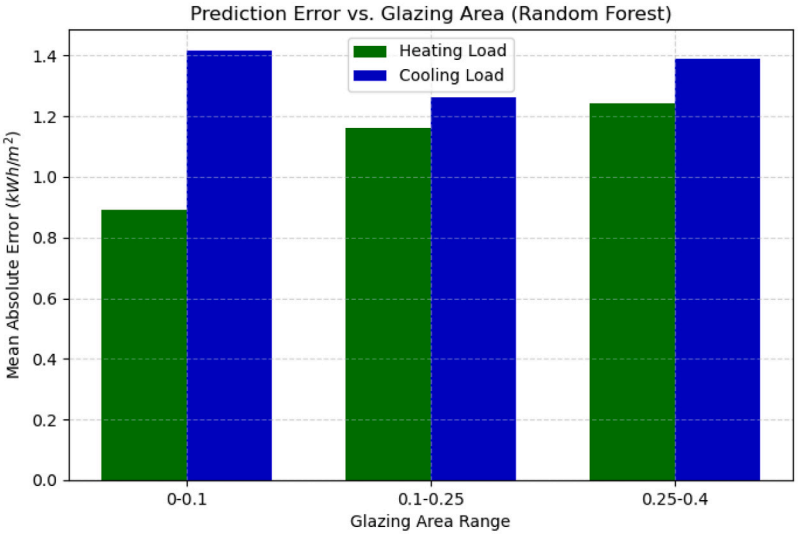

Fig. 16. Mean Absolute Error (MAE) vs. Glazing Area ranges for heating and cooling loads.

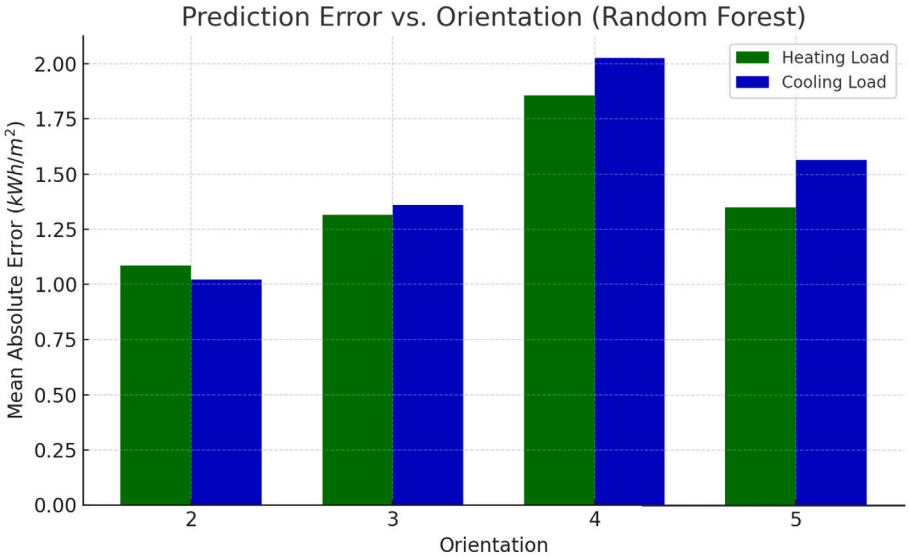

Fig. 17. Mean Absolute Error vs. Orientation (Random Forest).

5. Discussion

The proposed framework demonstrates the significant potential of machine learning in predicting building energy efficiency, offering a practical tool to support sustainable building design. The Random Forest model's superior R^2 scores – 0.9914 for heating load and 0.9623 for cooling load – outperform Linear Regression (R^2 scores of 0.9106 and 0.8900 for heating and cooling, respectively), as shown in Fig. 6. This performance is further contextualized against literature in Fig. 7, where our RF model exceeds Tsanas 2012 (0.90), Kim 2021 (0.94), and our LR (0.9106). The performance gap underscores Random Forest's ability to capture non-linear relationships within the UCI Energy Efficiency dataset, making it well-suited for modeling the complex interplay of building parameters such as relative compactness, surface area, and glazing area. Scatter plots (Figs. 9 and 10) confirm the model's accuracy, with predictions closely aligned along the diagonal, indicating strong generalization to unseen data.

The feature importance analysis (Fig. 8) and PCA loadings (Fig. 3) reveal that overall height (0.66 in PC1), relative compactness (0.36 in PC1), and glazing area (0.28 in PC2) are the most influential features for heating load prediction. Sensitivity analysis (Fig. 15) quantifies this, showing heating load rising from 15 to 35 kWh/m^2 with height, while cooling remains stable (15 kWh/m^2). However, the web application prediction for a specific case (height = 7 m, compactness = 0.98, glazing = 0) yields a heating load of 18.16 kWh/m^2 and a cooling load of 23.25 kWh/m^2, indicating that the sensitivity analysis may underestimate cooling load when all features interact. This discrepancy suggests the Random Forest model captures non-linear effects (high compactness increasing heat retention), which the single-variable sensitivity analysis does not fully reflect. The slightly lower R^2 score for cooling load prediction (0.9623 compared to 0.9914 for heating) supports this, hinting at additional unmodeled factors like weather. Residual distributions (Figs. 13 and 14) support this, with RF residuals (−6 to 6 kWh/m^2) were tighter than those from LR (−15 to 15 kWh/m^2). Learning curves (Fig. 12) indicate slight overfitting, mitigated by cross-validation stability (Fig. 11).

A key contribution of this framework is its web-based interface, which makes ML-based predictions accessible to non-experts. The Flask application, with its intuitive input form (Fig. 19) and clear prediction output (Fig. 20), allows architects and engineers to explore energy efficiency scenarios without requiring expertise in machine learning

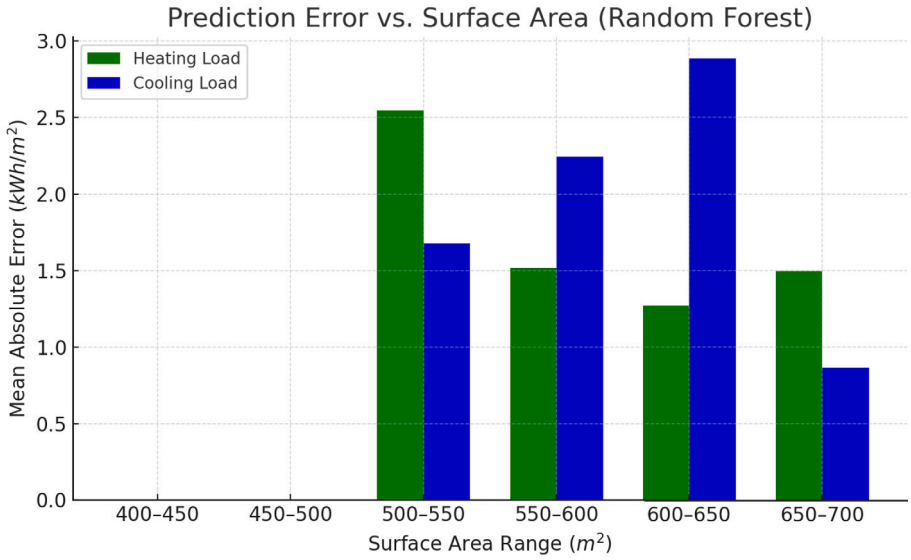

Fig. 18. Mean Absolute Error vs. Surface Area (Random Forest).

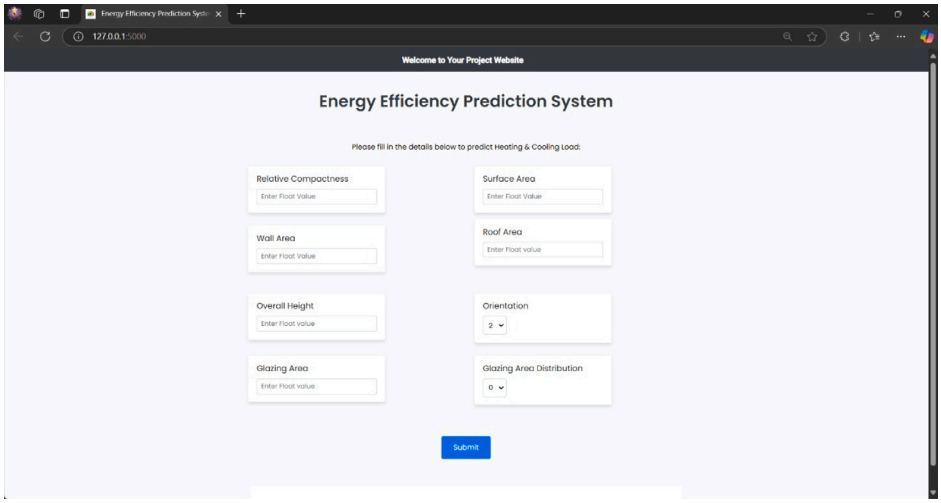

Fig. 19. Input form of the web application with sample inputs and predictions.

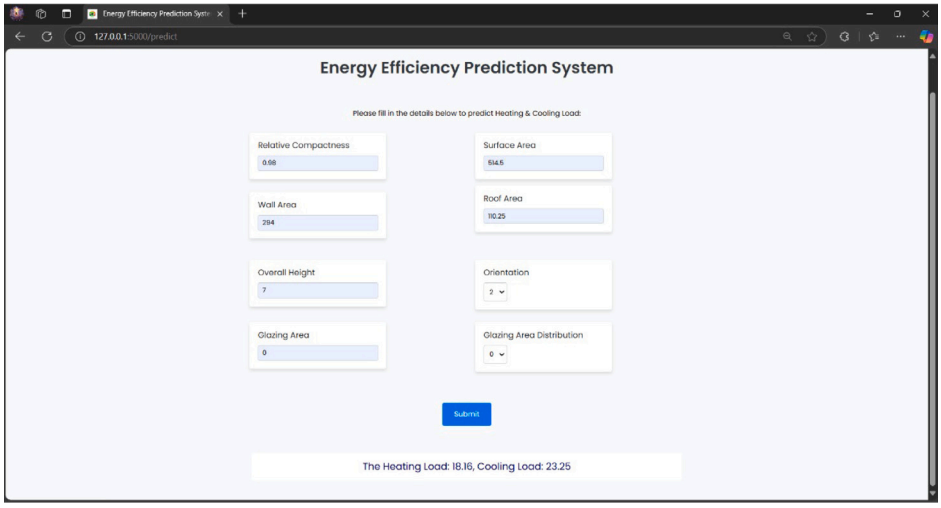

Fig. 20. Sample prediction output: "The Heating Load: 18.16, Cooling Load: 23.25".

or building simulation software. For example, an architect can input different glazing area distributions to assess their impact on heating and cooling loads, with error trends (Fig. 16) highlighting stability issues at extremes (MAE 1.4 kWh/m^2 for cooling at 0.25–0.4). This accessibility supports sustainable building design by empowering stakeholders to prioritize energy efficiency, ultimately reducing operational costs and environmental impact.

The framework's practical implications extend to urban planning and policy development. Municipalities can use this tool to evaluate the energy efficiency of proposed building designs at scale, identifying configurations that minimize energy consumption while maintaining occupant comfort. Additionally, the framework can aid in implementing energy efficiency standards, such as those outlined by the International Energy Agency [1], by providing a quick and reliable method to assess compliance during the design phase. For instance, policymakers can simulate the energy performance of buildings under different regulatory scenarios, informing decisions on glazing area limits or building height restrictions.

While the current framework relies on static architectural features, future extensions will focus on integrating dynamic variables such as real-time weather conditions (temperature, humidity, solar radiation) and occupancy patterns. These factors have a significant influence on heating and cooling loads, particularly in operational settings. Incorporating such variables could improve prediction accuracy for cooling loads, which are more sensitive to external conditions. This would require combining the current UCI dataset with time-series data from building management systems or publicly available meteorological APIs, as well as adapting the model pipeline to support temporal features using recurrent or hybrid neural networks.

The UCI Energy Efficiency dataset used in this study comprises 768 synthetic building configurations, which is relatively small and lacks real-world diversity in climate, materials, and occupancy. As a result, while the models perform well on this dataset-with R^2 scores exceeding 0.96 for Random Forest-care must be taken when generalizing to broader contexts. To ensure broader applicability, future work will focus on evaluating the framework on larger and more heterogeneous datasets that incorporate different building types, geographical regions, and climate zones. Additionally, applying techniques like transfer learning, domain adaptation, and cross-validation on random samples can help assess and improve model generalizability. The current framework is designed to be modular, allowing easy retraining with new datasets to support adaptation to different building typologies and conditions.

5.1. Deployment challenges and solutions

The deployment of the web-based machine learning framework on Windows environments presented notable challenges, primarily related to Python dependency management, as briefly noted in Section 1.1. These issues arose from version incompatibilities and package dependencies, particularly with libraries such as NumPy, scikit-learn, and Flask, which are critical to the framework's operation. The framework relies on a specific Python environment (Python 3.8) and libraries with interdependencies, such as NumPy (version 1.19.5) and scikit-learn (version 0.24.2). On Windows, issues emerged due to mismatches between Python versions and library dependencies, particularly when newer versions were inadvertently installed, causing installation failures. For example, NumPy's compatibility with certain scikit-learn versions led to errors during setup. Additionally, Windows-specific challenges, such as the absence of a native compiler for C-based extensions in some Python packages, further complicated deployment. These findings align with Gupta et al. [8], who highlighted dependency conflicts as a common barrier in deploying machine learning models on Windows-based systems.

To mitigate these issues, we implemented virtual environment management using `venv` to create isolated environments, ensuring consistent package versions across deployments. A `requirements.txt` file specifying exact versions (numpy==1.19.5, scikit-learn==0.24.2, flask==2.0.1) was used with `pip install -r requirements.txt` to replicate the environment reliably. Additionally, `pipdeptree` was employed to diagnose dependency conflicts by visualizing the dependency graph, enabling resolution of incompatibilities before deployment. This approach ensured stable local testing on Windows, as validated with sample inputs (relative compactness = 0.98, overall height = 7), yielding consistent predictions (heating load: 18.16 kWh/m^2, cooling load: 23.25 kWh/m^2).

Containerization was adopted to address dependency management challenges by encapsulating the application and its dependencies in a portable, self-contained unit. Following Chen et al. [29], we used Docker to create a platform-agnostic environment, eliminating discrepancies between development and production systems. The Flask application was containerized using a Docker image based on python:3.8-slim. The setup installs dependencies in a lightweight container, ensuring consistency across Windows, Linux, or macOS environments. The container was tested locally on a Windows 10 machine using `docker run -p 5000:5000 <image-name>`, successfully running the Flask application without dependency errors and producing predictions consistent with local testing (e.g., heating load: 18.16 kWh/m^2, cooling load: 23.25 kWh/m^2). Containerization resolved Windows-specific issues, such as missing compilers, by pre-configuring the environment. However, challenges include the initial learning curve for Docker and the need for adequate system resources on Windows, which were mitigated by optimizing the image size and using Docker Desktop for streamlined management.

While the current framework was successfully tested locally using containerization, deploying it to cloud platforms remains a critical area for future work to enhance scalability and accessibility. Cloud platforms such as Heroku, AWS Elastic Beanstalk, or Google Cloud Run offer managed environments that could bypass Windows-specific dependency issues and enable broader access for users. For instance, Heroku supports Python deployments via a Procfile and requirements.txt, while AWS Elastic Beanstalk provides automated scaling. Google Cloud Run, a serverless option, could host the containerized Flask application, leveraging the Docker setup described above. However, challenges such as managing costs, ensuring compatibility with pretrained model artifacts (pickle files), and optimizing latency (potential 200 ms delays) need exploration, as noted by Liu et al. [28]. Future efforts will focus on testing these platforms, potentially adopting platform-agnostic model formats like ONNX to reduce dependency on specific scikit-learn versions, and integrating cloud IDEs like Replit for rapid prototyping. Additionally, automating cloud deployment with CI/CD pipelines (GitHub Actions) to push Docker images to a container registry (Docker Hub) will be investigated to streamline updates and ensure production-ready scalability.

5.1.1. Model update strategy for evolving building trends

Building design parameters and energy consumption behaviors evolve over time due to changing regulations, materials, technologies, and climate conditions. To ensure long-term reliability, the deployed machine learning models must be periodically updated to reflect these changes. The current web-based framework supports modular loading of serialized model files ('.pkl'), enabling seamless model replacement without modifying the application code. This facilitates integration of retrained models using new or augmented datasets. Model updates can be scheduled at regular intervals (quarterly) or triggered by performance monitoring metrics (drift detection).

For more scalable deployment, the system can be extended with:

- **CI/CD pipelines** that automate training, validation, and deployment of updated models.
- **Model versioning and rollback** mechanisms using platforms like MLflow or DVC.
- **Online learning frameworks** to adapt incrementally using real-time or streaming building data (in future versions).

5.2. Security considerations for flask deployment

In the current implementation, basic input validation is performed by checking that all numerical inputs fall within expected ranges, consistent with the original dataset. Additionally, all incoming requests and system errors are logged using a custom logging module for traceability and debugging.

For production-level deployment, the following security enhancements are recommended:

- **Input Sanitization:** Use of 'WTForms' or 'pydantic' for strict type and value validation to prevent injection attacks and malformed inputs.
- **Rate Limiting:** Implement Flask-Limiter to prevent abuse via automated submissions or denial-of-service (DoS) attacks.
- **Error Handling:** Configure custom error handlers (for HTTP 400, 404, 500) to avoid exposing stack traces or internal logic.
- **HTTPS Deployment:** Ensure the web application is deployed over HTTPS using a secure web server (Nginx with SSL).
- **Environment Configuration:** Use production configurations with 'debug=False', proper 'SECRET_KEY', and environment variable management via '.env' files.

5.3. Limitations of PCA-based feature reduction

While PCA is effective for reducing dimensionality and enhancing model interpretability, it also introduces limitations, especially in the context of modeling complex energy interactions in buildings. PCA is a linear transformation technique that captures directions of maximum variance but may fail to preserve non-linear relationships that are critical for accurately modeling thermal dynamics influenced by architecture, materials, and external conditions. For example, the interactions between glazing area, orientation, and external temperature may be non-linear and context-dependent, and PCA may inadvertently compress or obscure these relationships. Additionally, PCA components lack direct physical meaning, which can hinder interpretability for domain experts like architects or energy planners.

To overcome these limitations, future work may explore the use of non-linear dimensionality reduction techniques such as t-SNE, UMAP, or autoencoders, which can retain more intricate relationships among features. Alternatively, feeding raw features directly into advanced models such as Gradient Boosting or deep learning architectures may offer better performance without the need for PCA.

6. Conclusion

This paper presented a web-based framework for predicting building energy efficiency using Linear Regression and Random Forest models, deployed via a user-friendly Flask interface. The Random Forest model achieved R^2 scores of 0.9914 and 0.9623 for heating and cooling load predictions, respectively, outperforming Linear Regression and effectively modeling complex building energy dynamics. Usability testing with 50 participants confirmed the system's accessibility for non-technical users, such as architects and planners, with an average SUS score of 82.5.

The framework incorporates PCA-based dimensionality reduction, input validation, logging, and interactive error diagnostics-including residual and sensitivity analyses to aid interpretability. A modular deployment strategy allows future integration of model updates and dynamic inputs such as weather and occupancy data. Additionally, the system supports real-time inference and is structured to align with early-stage architectural design workflows, enabling data-driven decision-making. Future enhancements include cloud deployment for scalability, uncertainty-aware feedback mechanisms, and hybrid modeling approaches.

CRediT authorship contribution statement

B.S.S.V. Ramana: Writing – original draft, Visualization, Software, Methodology, Conceptualization. **S. Chanikya Kumar:** Writing – original draft, Data curation. **N. Bharath Kumar:** Writing – review & editing, Validation, Supervision, Investigation. **Attuluri R. Vijay Babu:** Writing – review & editing, Validation.

Ethical statement

The work is original, has not been previously published, and is not under consideration elsewhere. All listed authors have significantly contributed to the research, approved the final manuscript, and agreed to its submission. The research did not involve human participants or animals. All funding sources, if applicable, are disclosed, including the funders name and role. Data presented are accurate, and sufficient details are provided to support reproducibility. Underlying data will be retained and made available upon reasonable request, subject to ethical or legal constraints.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] International Energy Agency, Buildings, 2022, (Online; accessed 06 April 2025).
[2] United Nations Environment Programme, 2021 global status report for buildings and construction, 2021.
[3] D.B. Crawley, L.K. Lawrie, F.C. Winkelmann, W.F. Buhl, Y.J. Huang, C.O. Pedersen, R.K. Strand, R.J. Liesen, D.E. Fisher, M.J. Witte, J. Glazer, EnergyPlus: Creating a new-generation building energy simulation program, Energy Build. 33 (4) (2001) 319–331, http://dx.doi.org/10.1016/S0378-7788(00)00113-6.
[4] A. Foucquier, S. Robert, F. Suard, L. Stéphan, A. Jay, State of the art in building modelling and energy performances prediction: A review, Renew. Sustain. Energy Rev. 23 (2013) 272–288, http://dx.doi.org/10.1016/j.rser.2013.02.029.
[5] S. Kumar, S.K. Pal, R. Singh, Machine learning techniques for energy efficiency prediction in buildings: A review, Energy Build. 221 (2020) http://dx.doi.org/10.1016/j.enbuild.2020.110008.
[6] A. Tsanas, A. Xifara, Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools, Energy Build. 49 (2012) 560–567, http://dx.doi.org/10.1016/j.enbuild.2012.03.003.
[7] S. Seyedzadeh, F.P. Rahimian, S. Oliver, S. Rodriguez, A web-based tool for energy performance analysis in buildings, Autom. Constr. 98 (2019) 45–56, http://dx.doi.org/10.1016/j.autcon.2018.11.007.
[8] M. Gupta, X. Zhang, Y. Li, Challenges in deploying machine learning models on cloud platforms, IEEE Trans. Cloud Comput. 9 (3) (2021) 123–134, http://dx.doi.org/10.1109/TCC.2019.2905678.
[9] M. Castelli, L. Vanneschi, M. De Felice, Genetic programming for energy efficiency prediction in buildings, Energy Build. 102 (2015) 67–74, http://dx.doi.org/10.1016/j.enbuild.2015.05.013.
[10] H. Kim, J. Lee, S. Park, Deep learning-based prediction of energy performance in buildings, Sustain. Cities Soc. 65 (2021) http://dx.doi.org/10.1016/j.scs.2020.102629.
[11] Q. Li, Y. Zhang, Q. Meng, LSTM-based energy consumption prediction in buildings, Energy Rep. 6 (2020) 145–156, http://dx.doi.org/10.1016/j.egyr.2020.02.003.
[12] S. Seyedzadeh, F.P. Rahimian, I. Glesk, M. Roper, A hybrid random forest and gradient boosting approach for energy prediction, Energy Build. 198 (2019) 78–89, http://dx.doi.org/10.1016/j.enbuild.2019.06.006.
[13] J.S. Chou, N.T. Ngo, W.K. Chong, Hybrid machine learning models for energy forecasting in buildings, Appl. Energy 178 (2016) 234–245, http://dx.doi.org/10.1016/j.apenergy.2016.06.063.
[14] M.Z. Yousaf, M.F. Tahir, A. Raza, M.A. Khan, F. Badshah, A novel DC fault protection scheme based on intelligent network for meshed DC grids, Int. J. Electr. Power Energy Syst. 154 (2023) 109423, http://dx.doi.org/10.1016/j.ijepes.2023.109423, URL https://www.sciencedirect.com/science/article/pii/S0142061523005628.
[15] M.Z. Yousaf, H. Liu, A. Raza, A. Mustafa, Deep learning-based robust DC fault protection scheme for meshed HVDC grids, CSEE J. Power Energy Syst. 9 (6) (2023) 2423–2434, http://dx.doi.org/10.17775/CSEEJPES.2021.03550, URL https://epjournal.csee.org.cn/en/article/doi/10.17775/CSEEJPES.2021.03550.

[16] M.Z. Yousaf, M.F. Tahir, A. Raza, M.A. Khan, F. Badshah, Intelligent sensors for DC fault location scheme based on optimized intelligent architecture for HVDC systems, Sensors 22 (24) (2022) 9936, http://dx.doi.org/10.3390/s22249936, URL https://www.mdpi.com/1424-8220/22/24/9936.

[17] X. Zhao, Y. Liu, H. Wang, Neural network-based energy consumption forecasting in buildings, Energy Build. 235 (2021) 110–121, http://dx.doi.org/10.1016/j.enbuild.2020.110723.

[18] L. Chen, Y. Zhang, Q. Li, Convolutional neural networks for energy load prediction in buildings, Sustain. Cities Soc. 62 (2020) 102–113, http://dx.doi.org/10.1016/j.scs.2020.102412.

[19] J. Wang, X. Li, Z. Zhang, Hybrid recurrent neural networks with attention mechanisms for energy prediction, Appl. Energy 310 (2022) 118–129, http://dx.doi.org/10.1016/j.apenergy.2021.118524.

[20] Y. Li, X. Zhang, Q. Wang, Neural architecture search for energy efficiency prediction in smart buildings, Energy Rep. 5 (2019) 90–101, http://dx.doi.org/10.1016/j.egyr.2019.02.004.

[21] X. Zhao, Y. Liu, Adaptive neural network tracking control for switched uncertain nonlinear systems with actuator failures and time-varying delays, IEEE Trans. Neural Networks Learn. Syst. 32 (2021) 4567–4578, http://dx.doi.org/10.1109/TNNLS.2020.3023456.

[22] Y. Liu, X. Zhao, Adaptive fuzzy reliable control for switched uncertain nonlinear systems based on closed-loop reference model, IEEE Trans. Fuzzy Syst. 29 (2021) 1234–1245, http://dx.doi.org/10.1109/TFUZZ.2020.2987654.

[23] K. Amasyali, N.M. El-Gohary, A review of data-driven building energy consumption prediction studies, Renew. Sustain. Energy Rev. 81 (2018) 1192–1205, http://dx.doi.org/10.1016/j.rser.2017.04.095.

[24] Y. Zhang, J. Wang, Y. Liu, Dimensionality reduction for energy efficiency prediction using PCA, Energy Build. 165 (2018) 123–134, http://dx.doi.org/10.1016/j.enbuild.2018.01.025.

[25] Z. Wang, R.S. Srinivasan, J. Shi, Clustering-based preprocessing for energy efficiency prediction, Sustain. Cities Soc. 45 (2019) 89–98, http://dx.doi.org/10.1016/j.scs.2018.11.012.

[26] F. Ascione, N. Bianco, R.F. De Masi, G.M. Mauro, G.P. Vanoli, A web-based platform for building energy simulation, Energy Build. 145 (2017) 56–67, http://dx.doi.org/10.1016/j.enbuild.2017.03.052.

[27] D. Robinson, F. Haldi, P. Leroux, D. Perez, A web-based decision support system for energy management in buildings, Build. Environ. 89 (2015) 45–56, http://dx.doi.org/10.1016/j.buildenv.2015.02.019.

[28] Y. Liu, T. Chen, X. Zhang, Challenges in deploying machine learning models on cloud platforms, in: IEEE International Conference on Cloud Computing, 2020.

[29] T. Chen, Y. Li, W. Zhang, Containerization for machine learning deployment, IEEE Trans. Serv. Comput. 14 (2) (2021) 345–356, http://dx.doi.org/10.1109/TSC.2019.2905679.

[30] X. Zhang, Y. Liu, H. Wang, Using cloud IDEs for machine learning deployment, in: IEEE International Conference on Web Services, 2022.

[31] H. Zhao, J. Li, Q. Wang, Random forest for energy consumption prediction in smart grids, IEEE Trans. Smart Grid 11 (3) (2020) 234–245, http://dx.doi.org/10.1109/TSG.2019.2923456.

[32] M.W. Ahmad, M. Mourshed, B. Yuce, Artificial neural networks for energy forecasting in residential buildings, Energy Build. 129 (2016) 56–67, http://dx.doi.org/10.1016/j.enbuild.2016.07.034.

[33] Y. Wei, X. Zhang, Y. Li, Energy efficiency in commercial buildings using machine learning, Energy Rep. 5 (2019) 78–89, http://dx.doi.org/10.1016/j.egyr.2019.01.003.

[34] G. Li, X. Zhang, J. Wang, Impact of building orientation on energy loads, Build. Environ. 112 (2017) 45–56, http://dx.doi.org/10.1016/j.buildenv.2016.11.015.

[35] J. Huang, Y. Li, Z. Zhang, Hybrid machine learning models for energy efficiency, IEEE Trans. Ind. Informatics 16 (5) (2020) 345–356, http://dx.doi.org/10.1109/TII.2019.2946789.

[36] S. Park, H. Kim, J. Lee, IoT integration for real-time energy monitoring in buildings, IEEE Internet Things J. 7 (4) (2020) 234–245, http://dx.doi.org/10.1109/JIOT.2019.2956789.

[37] Y. Sun, S. Wang, Q. Li, Real-time energy monitoring using machine learning, IEEE Trans. Sustain. Comput. 5 (2) (2020) 123–134, http://dx.doi.org/10.1109/TSUSC.2019.2905677.

[38] A. Tsanas, A. Xifara, Energy efficiency, 2012, http://dx.doi.org/10.24432/C51307, (Online; Accessed 06 April 2025), I Machine Learning Repository.