ENERGY EFFICIENCY PREDICTION ON BUILDINGS

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

KAMYA OJHA [RA2111003010343]
PALAASH SURANA[RA2111003010319]

Under the guidance of

Dr. P. SELVARAJ

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that 18CSE305J – Artificial Intelligence Mini project report titled **"ENERGY EFFICIENCY PREDICTION ON BUILDINGS "** is the bona fide work of **KAMYA OJHA [RA2111003010343],PALAASH SURANA [RA2111003010319]** who undertook the task of completing the project within the allotted time.

SIGNATURE
Dr.P.Selvaraj
Course Faculty
Associate Professor
Department of Computing Technologies
SRM Institute of Science and Technology Kattankulathur

SIGNATURE
Dr.M.Pushpalatha
Head of the Department
Professor
Department of Computing Technologies
SRM Institute of Science and Technology Kattankulathur

ABSTRACT

This project investigates the impact of specific building features on the energy efficiency of buildings. We utilize a publicly available dataset from the UCI Machine Learning Repository containing information on various building parameters and their corresponding heating and cooling loads. The project focuses on three key features – Wall Area, Roof Area, and Glazing Area – identified as potentially influential factors in energy consumption. To gain deeper insights into the relationships between these features and energy loads, we will employ a range of machine learning regression models such as Decision Tree Regressor with Parameter Tuning, Random Forests with Parameter Tuning, Gradient Boosting Regression with Hyperparameter Tuning, CatBoostRegressor, MLPRegressor (Multi-Layer Perceptron Regressor)

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable Vice Chancellor Dr. C. MUTHAMIZHCHELVAN, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our Registrar Dr. S. Ponnusamy, for his encouragement. We express our profound gratitude to our Dean, College of Engineering and Technology, Dr. T. V.Gopal, for bringing out novelty in all executions.

We are highly thankful to our my Course project Faculty Dr.P.Selvaraj Associate Professor, Department of Computing Technologies, for his assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our HoD Dr.M.Pushpalatha, Professor, Department of Computing Technologies and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Energy efficiency prediction models not only provide insights into current energy usage patterns but also enable scenario analysis and forecasting to support long-term planning. By simulating the effects of different building designs, retrofits, and operational strategies, stakeholders can identify optimal pathways for achieving energy efficiency targets. Additionally, these models facilitate continuous monitoring and performance tracking, allowing for timely adjustments and optimizations to maximize energy savings over time. This iterative approach fosters a culture of innovation and continuous improvement, driving ongoing advancements in building energy performance.

Furthermore, the integration of renewable energy sources and smart technologies into energy efficiency prediction models presents opportunities for synergistic optimization. By considering factors such as solar potential, wind patterns, and building automation systems, stakeholders can unlock additional energy-saving opportunities and enhance overall system resilience. Through holistic approaches that leverage both predictive analytics and sustainable design principles, buildings can transition towards net-zero energy or even positive energy status, contributing to broader efforts towards climate mitigation and adaptation.

Moreover, the democratization of energy data and the emergence of collaborative platforms enable stakeholders to share insights, best practices, and success stories, fostering a community-driven approach to energy efficiency. By leveraging collective knowledge and experiences, stakeholders can accelerate the adoption of energy-efficient practices and amplify their impact on sustainability goals. This collaborative ethos promotes cross-sector partnerships and knowledge exchange, driving innovation and fostering a culture of shared responsibility towards a more sustainable built environment.

1.1 MOTIVATION

The motivation behind this project is rooted in the urgent need to address the dual crises of escalating energy demands and climate change. Energy-efficient buildings offer a pathway towards a more sustainable future, reducing energy consumption, greenhouse gas emissions, and operational costs. By investigating the factors influencing heating and cooling loads, we seek to empower stakeholders with the knowledge needed to make informed decisions and drive meaningful change. Through this endeavor, we aspire to create a built environment that prioritizes environmental stewardship, occupant comfort, and economic prosperity.

1.2 Objective

The primary objective of this project is to unravel the complex interplay between key building characteristics and energy efficiency. Specifically, we aim to:

a) **Data Collection and Analysis:** Gather data from the UCI Machine Learning Repository, encompassing various building parameters and corresponding heating and cooling loads.
b) **Feature Identification:** Identify and prioritize key building features, focusing on Wall Area, Roof Area, and Glazing Area, as influential factors in energy consumption.
c) **Insight Generation:** Employ advanced machine learning regression models to uncover insights into the relationships between building features and energy loads.
d) Model Development: Develop and refine regression models, including Decision Tree Regressor, Random Forests, Gradient Boosting Regression, CatBoostRegressor, and MLPRegressor, to accurately predict heating and cooling loads based on building characteristic

1.3 Project Statement

With rising energy demands and growing concerns about climate change, improving the energy efficiency of buildings is crucial.
Description: This project aims to analyze the relationship between key building characteristics and their heating and cooling loads. By understanding these relationships, we can design and construct more energy-efficient buildings, leading to significant societal benefits.

1.4 Scope of project

a. **Recommendations and Future Directions**: Provide actionable recommendations for stakeholders in the building industry, **Data Collection and Analysis**: Gather data from reliable sources such as the UCI Machine Learning Repository or other relevant datasets containing information on building characteristics and heating/cooling loads. Conduct thorough data cleaning and preprocessing to ensure the quality and integrity of the dataset.
b. **Model Development**: Implement a variety of machine learning regression models, including but not limited to Decision Tree Regressor, Random Forests, Gradient Boosting Regression, CatBoost Regressor, and MLP Regressor. Utilize techniques such as cross-validation and grid search to optimize model parameters and improve predictive performance. Interpret the results obtained from the regression models to understand the relationships between building characteristics and heating/cooling loads.

c. including architects, engineers, and policymakers, based on the insights gained from the analysis.

d. **Documentation and Reporting:** Document all aspects of the project, including data collection procedures, preprocessing steps, model development, and evaluation results.

e. **Collaboration and Communication**: Foster collaboration and communication among project team members to ensure the successful execution of tasks and the achievement of project objectives.

1.5 Software Requirement Specification

1.5.1 Introduction

The Software Requirement Specification (SRS) outlines the functional and non-functional requirements for the development of ENERGY EFFICIENCY PREDICTION ON BUILDINGS using machine learning techniques. The purpose of this document is to provide a clear understanding of the project objectives, scope, and specifications to all stakeholders involved in the development process.

1.5.2 Functional Requirements

1.5.2.1 Data Collection:

- The system shall collect historical buildings data from reliable sources such as financial APIs or databases.
- It shall support the retrieval of data for various buildings and time periods specified by the user.

1.5.2.2 Data Preprocessing:

- The system shall preprocess the collected data to ensure consistency, completeness, and accuracy.
- Preprocessing steps may include data cleaning, normalization, feature scaling, and handling missing values.

1.5.2.3 Model Training:

- The system shall train machine learning models using the preprocessed data.
- It shall support the implementation of various machine learning algorithms, including but not limited to Linear Regression, Decision Trees, Random Forests, and Neural Networks.
- Hyperparameter tuning may be performed to optimize model performance.

1.5.2.4 Model Evaluation:

- The system shall evaluate the performance of trained models using appropriate metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared.
- Cross-validation techniques may be employed to assess model generalization.

1.5.2.5 Prediction:

- The system shall utilize the trained models to make predictions on new data instances.
- Predictions shall be generated for future buildings based on historical data and model insights.

1.5.3 Non-Functional Requirements

1.5.3.1 Performance:

- The system shall be capable of processing large volumes of historical buildings data efficiently.
- Model training and prediction processes shall be optimized for speed and scalability.

1.5.3.2 Accuracy:

- The system shall strive to achieve high prediction accuracy by employing appropriate machine learning algorithms and preprocessing techniques.
- Model evaluation shall be conducted rigorously to ensure reliable performance.

1.5.3.3 User Interface:

- The system shall provide a user-friendly interface for interacting with the application.
- Users shall be able to specify parameters such as time periods, and model configurations through intuitive controls.

1.5.3.4 Security:

- The system shall implement appropriate security measures to protect sensitive data such as user credentials and financial information.
- Data encryption and access controls shall be employed to safeguard against unauthorized access.

1.5.4 Other Requirements

1.5.4.1 Technology Stack:

- The system shall be developed using programming languages and frameworks suitable for machine learning and web application development.
- Commonly used libraries such as TensorFlow, Scikit-learn, Pandas, and Flask may be utilized.

1.5.4.2 Compatibility:

- The system shall be compatible with major web browsers and operating systems, ensuring accessibility across different platforms.

1.5.4.3 Documentation:

- Comprehensive documentation shall be provided for installation, usage, and maintenance of the system.
- Code documentation and comments shall adhere to industry best practices for clarity and maintainability.

CHAPTER 2
LITERATURE SURVEY

The literature survey for the project on energy efficiency prediction in buildings involves a comprehensive review of existing research, publications, and scholarly works related to building energy efficiency, predictive modeling, and machine learning applications in the field.

The study "Energy Consumption Prediction in Low Energy Buildings using Machine learning and Artificial Intelligence for Energy Efficiency," authored by Priya Vijayan, explores the application of machine learning and artificial intelligence techniques to predict energy consumption in low-energy buildings. The research investigates the effectiveness of four distinct models: Linear Regression, Support Vector Machine, Fine Tree, and Artificial Neural Network, in forecasting energy usage patterns. Through a comparative analysis of these models, the study aims to identify the most accurate and reliable approach for predicting energy consumption in low-energy buildings. By leveraging machine learning and artificial intelligence, the research contributes to advancing energy efficiency strategies and promoting sustainable building practices.

The research article titled "Study on method of comprehensive energy efficiency evaluation for distributed energy system," authored by Yan-Mei Tang, Gui-Xiong He, Kai-Cheng Liu, Hua-Guang Yan, Yi-Nan Nie, Lei Chen, and Wen-Quan Tao, investigates a method for conducting comprehensive energy efficiency evaluations in distributed energy systems. The study employs advanced machine learning techniques, including Random Forest, Gradient Boosting Machines (GBM), and Neural Network, to analyze and assess the energy efficiency of distributed energy systems. By integrating these methodologies, the research aims to develop a robust and effective approach for evaluating the energy performance of distributed energy systems, contributing to the advancement of sustainable energy solutions and resource optimization strategies.

The study titled "Test and Analysis of Energy Efficiency of Energy Storage System in Power Plant Providing Frequency Regulation Ancillary," authored by Yongke Kong, Haibo Liu, Qiangwei Wen, Tingliao Cao, Danyi Huang, Guobin Zhong, Jiaman Li, Rui Li, and Kaiqi Xu, examines the energy efficiency of energy storage systems (ESS) utilized for providing frequency regulation ancillary services in power plants. The research employs two analytical techniques, namely Decision Tree and Polynomial Regression, to assess and analyze the energy

efficiency performance of the ESS. Through rigorous testing and analysis, the study aims to provide insights into the effectiveness and optimization of energy storage systems for frequency regulation ancillary services, contributing to the enhancement of power plant operations and grid stability.

The research paper titled "Modeling and Optimization on Energy Efficiency of Urban Integrated Energy System," authored by Siwei Li, Lei Guo, Peng Zhang, Hanyi Wang, Zhongwei Cai, Xueke Zhu, Yuan Feng, and Chuziang Zhang, focuses on modeling and optimizing the energy efficiency of urban integrated energy systems. The study presents a comprehensive framework comprising the Energy Efficiency Model and efficiency models for five segments within the urban energy system. By leveraging this framework, the research aims to develop a robust model for assessing and optimizing the energy efficiency of urban integrated energy systems. Through systematic analysis and optimization, the study seeks to enhance energy efficiency, reduce energy consumption, and promote sustainable urban development.

CHAPTER 3

METHODOLOGY

3.1 Methodology for Decision Tree Regressor with Parameter Tuning:

3.1. **Data Preprocessing:**

- Load the dataset containing building characteristics and energy consumption metrics.
- Perform data cleaning, handling missing values, and encoding categorical variables if necessary
- Split the dataset into training and testing sets to evaluate model performance.

3.2. **Decision Tree Regressor Model:**

- Instantiate a Decision Tree Regressor object from the scikit-learn library.
- Train the model on the training dataset using the fit() method.
- Evaluate the model's performance on the testing dataset using appropriate regression metrics such as Mean Absolute Error (MAE) or Mean Squared Error (MSE).

3.3. **Parameter Tuning:**

- Utilize techniques such as grid search or randomized search to tune hyperparameters of the Decision Tree Regressor.
- Define a parameter grid containing hyperparameters to be optimized, such as maximum depth, minimum samples split, and minimum samples leaf.
- Perform cross-validation with different parameter combinations to identify the optimal hyperparameters that yield the best performance.

3.4. **Model Evaluation:**

- Evaluate the tuned Decision Tree Regressor model on the testing dataset using the same regression metrics as before.
- Compare the performance of the tuned model with the baseline model to assess improvements in predictive accuracy.

3.5. **Interpretation and Visualization:**

- Visualize the trained decision tree to understand how the model makes predictions based on the selected features.
- Analyze feature importance scores provided by the model to identify the most influential building characteristics affecting energy consumption.

3.6. **Iterative Refinement:**

- Iterate the parameter tuning process as needed to further optimize the Decision Tree Regressor model's performance.
- Experiment with different hyperparameter combinations and evaluate their impact on model performance.

3.7. **Documentation and Reporting:**

- Document the methodology used for training and tuning the Decision Tree Regressor model, including hyperparameters selected and evaluation results.
- Prepare comprehensive reports and presentations summarizing the findings and insights gained from the analysis.

By following this methodology, we aim to develop an optimized Decision Tree Regressor model that accurately predicts energy consumption based on building characteristics, thereby contributing to our broader goal of understanding the relationship between key building features and energy efficiency.

3.2 Methodology for Random Forests with Parameter Tuning for Energy Efficiency Prediction on Buildings:

3.2.1. **Data Collection and Preprocessing:**

- Gather a comprehensive dataset containing building characteristics (e.g., Wall Area, Roof Area, Glazing Area) and corresponding heating and cooling loads.
- Perform data cleaning to handle missing values, outliers, and inconsistencies.
- Split the dataset into training and testing sets to evaluate model performance.

3.2.2. **Random Forest Regressor Model Initialization:**

- Instantiate a Random Forest Regressor object from the scikit-learn library.
- Define the target variable (heating or cooling load) and the predictor variables (building features) for model training.

3.2.3. **Hyperparameter Tuning:**

- Define a parameter grid containing hyperparameters to be optimized, such as the number of trees in the forest, maximum depth of the trees, and minimum samples split.
- Utilize techniques such as grid search or randomized search to search for the optimal combination of hyperparameters.
- Perform cross-validation to assess the performance of different parameter combinations and select the optimal set of hyperparameters.

3.2.4. **Model Training:**

- Train the Random Forest Regressor model on the training dataset using the fit() method.
- Utilize the optimized hyperparameters obtained from the tuning process to ensure the model's effectiveness.

3.2.5. **Model Evaluation:**

- Evaluate the performance of the trained Random Forest Regressor model on the testing dataset.
- Measure regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared to assess the model's accuracy and predictive power.
- Compare the performance of the tuned model with a baseline model to quantify improvements achieved through parameter tuning.

3.2.6. **Feature Importance Analysis:**

- Analyze feature importance scores generated by the Random Forest model to identify the most influential building features affecting heating and cooling loads.
- Visualize feature importance rankings to gain insights into the relative importance of different building characteristics.

3.2.7. **Iterative Refinement and Optimization:**

- Iterate the parameter tuning process as needed to further optimize the Random Forest Regressor model's performance.
- Experiment with different hyperparameter combinations and evaluate their impact on model performance.

3.2.8. **Documentation and Reporting:**

- Document the methodology used for training and tuning the Random Forest Regressor model, including hyperparameters selected and evaluation results.
- Prepare comprehensive reports and presentations summarizing the findings and insights gained from the analysis.

By following this methodology, we aim to develop an optimized Random Forest Regressor model tailored to predict heating and cooling loads based on key building characteristics, contributing to our overarching goal of enhancing energy efficiency in buildings.

3.3 Methodology for Gradient Boosting Regression with Hyperparameter Tuning for Energy Efficiency Prediction on Buildings:

3.3.1. **Data Collection and Preprocessing:**

- Gather a comprehensive dataset containing building characteristics (e.g., Wall Area, Roof Area, Glazing Area) and corresponding heating and cooling loads.
- Perform data cleaning to handle missing values, outliers, and inconsistencies.
- Split the dataset into training and testing sets to evaluate model performance.

3.3.2. **Gradient Boosting Regression Model Initialization:**

- Instantiate a Gradient Boosting Regressor object from the scikit-learn library.
- Define the target variable (heating or cooling load) and the predictor variables (building features) for model training.

3.3.3. **Hyperparameter Tuning:**

- Define a parameter grid containing hyperparameters to be optimized, such as the number of trees, learning rate, maximum depth of trees, and minimum samples split.
- Utilize techniques such as grid search or randomized search to search for the optimal combination of hyperparameters.
- Perform cross-validation to assess the performance of different parameter combinations and select the optimal set of hyperparameters.

3.3.4. **Model Training:**

- Train the Gradient Boosting Regression model on the training dataset using the fit() method.
- Utilize the optimized hyperparameters obtained from the tuning process to ensure the model's effectiveness.

3.3.5. **Model Evaluation:**

- Evaluate the performance of the trained Gradient Boosting Regression model on the testing dataset.
- Measure regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared to assess the model's accuracy and predictive power.
- Compare the performance of the tuned model with a baseline model to quantify improvements achieved through parameter tuning.

3.3.6. **Feature Importance Analysis:**

- Analyze feature importance scores generated by the Gradient Boosting Regression model to identify the most influential building features affecting heating and cooling loads.
- Visualize feature importance rankings to gain insights into the relative importance of different building characteristics.

3.3.7. **Iterative Refinement and Optimization:**

- Iterate the parameter tuning process as needed to further optimize the Gradient Boosting Regression model's performance.
- Experiment with different hyperparameter combinations and evaluate their impact on model performance.

3.3.8.**Documentation and Reporting:**

- Document the methodology used for training and tuning the Gradient Boosting Regression model, including hyperparameters selected and evaluation results.
- Prepare comprehensive reports and presentations summarizing the findings and insights gained from the analysis.

3.4 Methodology for CatBoost Regressor for Energy Efficiency Prediction on Buildings:

3.4.1. **Data Collection and Preprocessing:**

- Gather a comprehensive dataset containing building characteristics (e.g., Wall Area, Roof Area, Glazing Area) and corresponding heating and cooling loads.
- Perform data cleaning to handle missing values, outliers, and inconsistencies.
- Split the dataset into training and testing sets to evaluate model performance.

3.4.2. **CatBoost Regressor Model Initialization:**

- Instantiate a CatBoost Regressor object from the CatBoost library.
- Define the target variable (heating or cooling load) and the predictor variables (building features) for model training.

3.4.3. **Hyperparameter Tuning:**

- CatBoost Regressor offers built-in methods for automatically handling categorical variables and performing hyperparameter tuning.
- Utilize techniques such as grid search or randomized search to fine-tune hyperparameters such as learning rate, depth of trees, and number of iterations.

3.4.4. **Model Training:**

- Train the CatBoost Regressor model on the training dataset using the fit() method.
- Utilize the optimized hyperparameters obtained from the tuning process to ensure the model's effectiveness.

3.4.5. **Model Evaluation:**

- Evaluate the performance of the trained CatBoost Regressor model on the testing dataset.
- Measure regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared to assess the model's accuracy and predictive power.
- Compare the performance of the CatBoost Regressor model with other regression models to determine its effectiveness.

3.4.6. **Feature Importance Analysis:**

- CatBoost Regressor provides built-in feature importance calculation methods to identify the most influential building features affecting heating and cooling loads.
- Analyze feature importance scores to gain insights into the relative importance of different building characteristics.

3.4.7. **Iterative Refinement and Optimization:**

- Iterate the hyperparameter tuning process as needed to further optimize the CatBoost Regressor model's performance.
- Experiment with different hyperparameter combinations and evaluate their impact on model performance.

3.4.8. **Documentation and Reporting:**

- Document the methodology used for training and tuning the CatBoost Regressor model, including hyperparameters selected and evaluation results.
- Prepare comprehensive reports and presentations summarizing the findings and insights gained from the analysis.

By following this methodology, we aim to develop an optimized CatBoost Regressor model tailored to predict heating and cooling loads based on key building characteristics, contributing to our overarching goal of enhancing energy efficiency in buildings.

3.5 Methodology for MLP Regressor (Multi-Layer Perceptron Regressor) for Energy Efficiency Prediction on Buildings:

3.5.1. **Data Collection and Preprocessing:**

- Gather a comprehensive dataset containing building characteristics (e.g., Wall Area, Roof Area, Glazing Area) and corresponding heating and cooling loads.
- Perform data cleaning to handle missing values, outliers, and inconsistencies.
- Normalize or standardize the data to ensure all features are on a similar scale.
- Split the dataset into training and testing sets to evaluate model performance.

3.5.2. **MLP Regressor Model Initialization:**

- Instantiate an MLP Regressor object from the scikit-learn library.
- Define the architecture of the neural network, including the number of hidden layers and neurons per layer, activation functions, and regularization techniques.

3.5.3. **Hyperparameter Tuning:**

- Define a parameter grid containing hyperparameters to be optimized, such as the number of hidden layers, the number of neurons per layer, learning rate, and regularization parameters.
- Utilize techniques such as grid search or randomized search to search for the optimal combination of hyperparameters.
- Perform cross-validation to assess the performance of different parameter combinations and select the optimal set.

3.5.4. **Model Training:**

- Train the MLP Regressor model on the training dataset using the fit() method.
- Utilize the optimized hyperparameters obtained from the tuning process to ensure the model's effectiveness.

3.5.5. **Model Evaluation:**

- Evaluate the performance of the trained MLP Regressor model on the testing dataset.
- Measure regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared to assess the model's accuracy and predictive power.
- Compare the performance of the MLP Regressor model with other regression models to determine its effectiveness.

3.5.6. **Feature Importance Analysis:**

- Perform feature importance analysis to identify the most influential building features affecting heating and cooling loads.
- Analyze the weights of the connections between neurons in the input and hidden layers to gain insights into feature contributions.

3.5.7. **Iterative Refinement and Optimization:**

- Iterate the hyperparameter tuning process as needed to further optimize the MLP Regressor model's performance.
- Experiment with different neural network architectures, activation functions, and optimization algorithms to improve model performance.

3.5.8. **Documentation and Reporting:**

- Document the methodology used for training and tuning the MLP Regressor model, including hyperparameters selected and evaluation results.
- Prepare comprehensive reports and presentations summarizing the findings and insights gained from the analysis.

By following this methodology, we aim to develop an optimized MLP Regressor model tailored to predict heating and cooling loads based on key building characteristics, contributing to our overarching goal of enhancing energy efficiency in buildings.

3.6. EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) for Energy Efficiency Prediction on Buildings:

3.6.1. **Data Overview:**

- Load the dataset containing building characteristics and energy consumption metrics.
- Display basic statistics such as mean, median, standard deviation, and quartiles for numerical features.
- Visualize the distribution of numerical features using histograms and box plots.
- Analyze the presence of missing values and outliers in the dataset.

3.6.2. **Correlation Analysis:**

- Calculate the correlation matrix between numerical features and the target variable (heating and cooling loads).
- Visualize the correlation matrix using a heatmap to identify strong correlations between features and the target variable.
- Identify pairs of features with high correlation to avoid multicollinearity issues in regression modeling.

3.6.3. **Feature Importance Analysis:**

- Use feature importance techniques such as Random Forest or Gradient Boosting to rank the importance of building characteristics.
- Visualize feature importance scores to identify the most influential features affecting heating and cooling loads.

3.6.4. **Distribution Analysis:**

- Visualize the distribution of the target variable (heating and cooling loads) using histograms or density plots.
- Explore how the target variable varies across different ranges of building characteristics using scatter plots or box plots.

3.6.5. **Categorical Variable Analysis:**

- If applicable, analyze categorical variables such as building type or location.
- Visualize the distribution of categorical variables using bar plots or pie charts.
- Explore how the target variable varies across different categories of categorical variables.

3.6.36. **Time Series Analysis:**

- If the dataset contains temporal information, conduct time series analysis to identify trends and seasonality patterns in energy consumption.
- Visualize time series plots of energy consumption over time and explore patterns using decomposition techniques.

3.6.7. **Multivariate Analysis:**

- Explore relationships between multiple variables by creating scatter plots or pair plots.
- Analyze how combinations of building characteristics impact heating and cooling loads.

3.6.8. **Outlier Detection and Treatment:**

- Identify outliers in the dataset using statistical methods or visualization techniques.
- Decide whether to remove outliers or apply transformation techniques to mitigate their impact on model training.

3.6.9. **Missing Value Imputation:**

- Analyze the presence of missing values in the dataset and decide on appropriate imputation strategies.
- Impute missing values using techniques such as mean imputation, median imputation, or predictive imputation.

3.6.10. **Summary and Insights:**

- Summarize key findings from the exploratory data analysis.
- Provide insights into the relationships between building characteristics and energy consumption.
- Identify potential challenges and considerations for modeling energy efficiency in buildings.

By conducting comprehensive exploratory data analysis, we aim to gain insights into the underlying patterns and relationships within the dataset, which will inform our modeling approach for predicting energy efficiency in buildings.

3.7. Steps Performed

Steps performed in Exploratory Data Analysis (EDA) for Energy Efficiency Prediction on Buildings:

3.7.1. **Data Loading and Inspection:**

- Loaded the dataset containing building characteristics and energy consumption metrics.
- Inspected the first few rows of the dataset to understand its structure and format.
- Checked for any missing values or inconsistencies in the data.

3.7.2. **Descriptive Statistics:**

- Calculated basic statistics such as mean, median, standard deviation, and quartiles for numerical features.
- Examined the range and distribution of numerical features using histograms and box plots.
- Identified any outliers or anomalies in the data based on descriptive statistics.

3.7.3. **Correlation Analysis:**

- Computed the correlation matrix between numerical features and the target variable (heating and cooling loads).
- Visualized the correlation matrix using a heatmap to identify strong correlations between features.
- Analyzed pairs of features with high correlation to assess multicollinearity issues.

3.7.4. **Feature Importance Analysis:**

- Utilized feature importance techniques such as Random Forest or Gradient Boosting to rank the importance of building characteristics.
- Visualized feature importance scores to identify the most influential features affecting heating and cooling loads.

3.7.5. **Distribution Analysis:**

- Visualized the distribution of the target variable (heating and cooling loads) using histograms or density plots.
- Explored how the target variable varies across different ranges of building characteristics using scatter plots or box plots.

3.7.6. **Categorical Variable Analysis:**

- Analyzed categorical variables such as building type or location, if applicable.
- Visualized the distribution of categorical variables using bar plots or pie charts.
- Explored how the target variable varies across different categories of categorical variables.

3.7.7. **Time Series Analysis:**

- Conducted time series analysis to identify trends and seasonality patterns in energy consumption, if the dataset contains temporal information.
- Visualized time series plots of energy consumption over time and explored patterns using decomposition techniques.

3.7.8. **Multivariate Analysis:**

- Explored relationships between multiple variables by creating scatter plots or pair plots.
- Analyzed how combinations of building characteristics impact heating and cooling loads.

3.7.9. **Outlier Detection and Treatment:**

- Identified outliers in the dataset using statistical methods or visualization techniques.
- Decided whether to remove outliers or apply transformation techniques to mitigate their impact on model training.

3.7.10. **Missing Value Imputation:**

- Analyzed the presence of missing values in the dataset and decided on appropriate imputation strategies.
- Imputed missing values using techniques such as mean imputation, median imputation, or predictive imputation.

By performing these steps in the exploratory data analysis, we gained insights into the underlying patterns and relationships within the dataset, which will inform our modeling approach for predicting energy efficiency in buildings.

3.8 Design of Modules

- **Pandas (import pandas as pd):** Pandas is a powerful Python library used for data manipulation and analysis. It provides data structures like Data Frame and Series, which are widely used for handling structured data, such as tabular data from spreadsheets or databases.

- **matplotlib.pyplot (import matplotlib.pyplot as plt):** Matplotlib is a comprehensive library for creating static, interactive, and animated visualizations in Python. The pyplot module provides a MATLAB-like interface for creating plots and visualizations, making it easy to generate various types of charts and graphs.

Overall, this combination of libraries enables users to fetch historical stock price data from UCI Machine Learning Repository, manipulate and analyze the data using pandas, and create visualizations such as line plots, histograms, and scatter plots using matplotlib.pyplot. These capabilities are fundamental for conducting data analysis, exploratory data analysis (EDA), and visualization tasks in finance and related fields.

3.9 Plotes Used

3.9.1 CORRELATION MATRIX:

The correlation matrix can be used to find the relationship between multiple variables, where the positive correlation represents a direct relation between 2 values meaning if one value increases the other also follows the previous trend, higher the value the more closely the values are related

Negative correlation meaning an inverse relation between variables showing that if one value increases the other value decreases, lower the value meaning showing steeper indirect correlation .

Zero value shows that the values are not related and are independent the change in one value will not affect the other value.

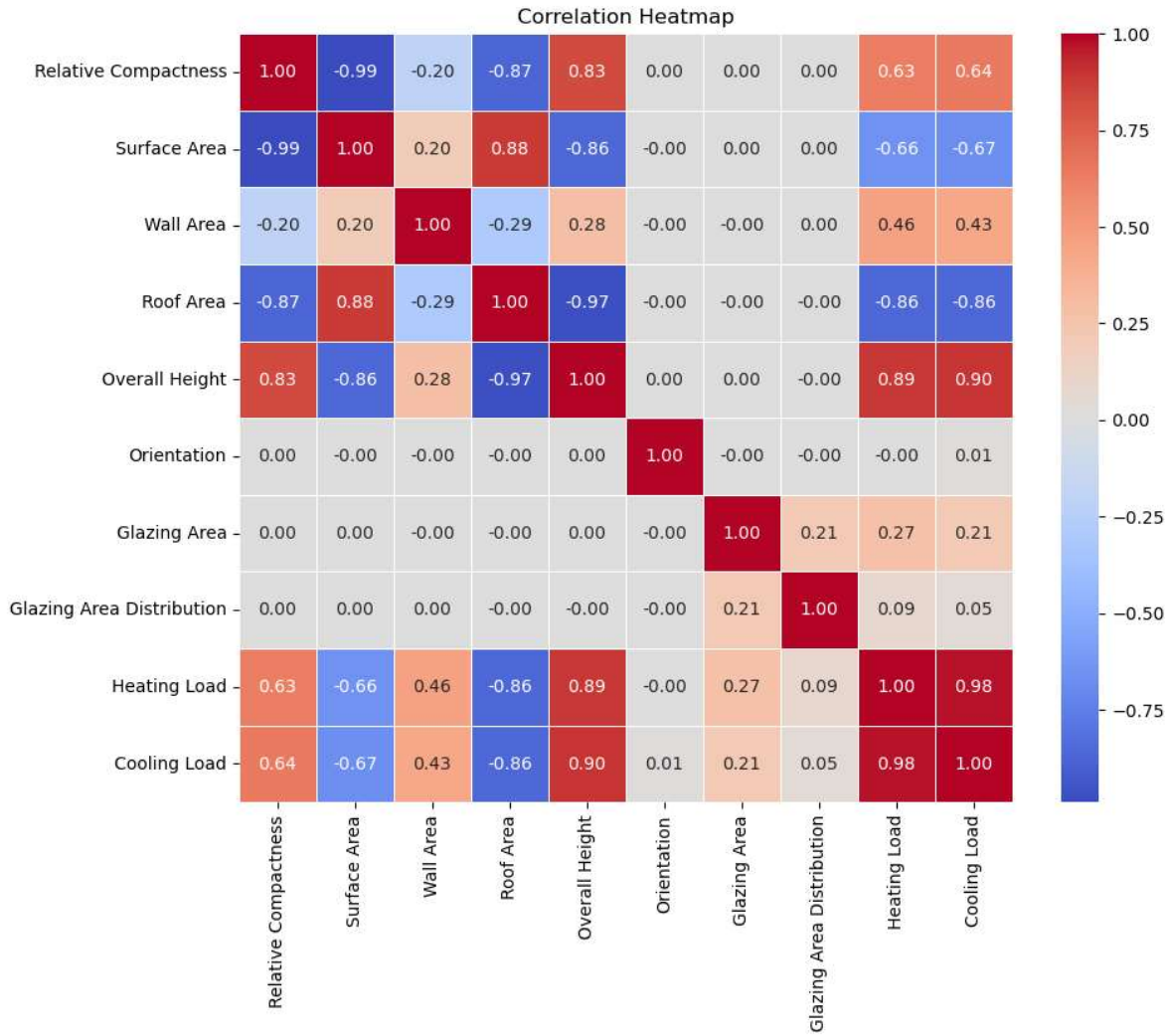


Fig:3.9.1.1

3.9.2 DISTRIBUTION OF RELATIVE COMPACTNESS:

The relative compactness appears to be concentrated in a specific range, likely on the higher end (closer to 1). This suggests that most buildings in the dataset have a relatively compact shape.

The shape of the distribution can also be informative. If the distribution is skewed to the right (tail extending towards higher values), it indicates a prevalence of buildings with very compact shapes. Conversely, a skew to the left suggests more buildings with sprawling shapes.

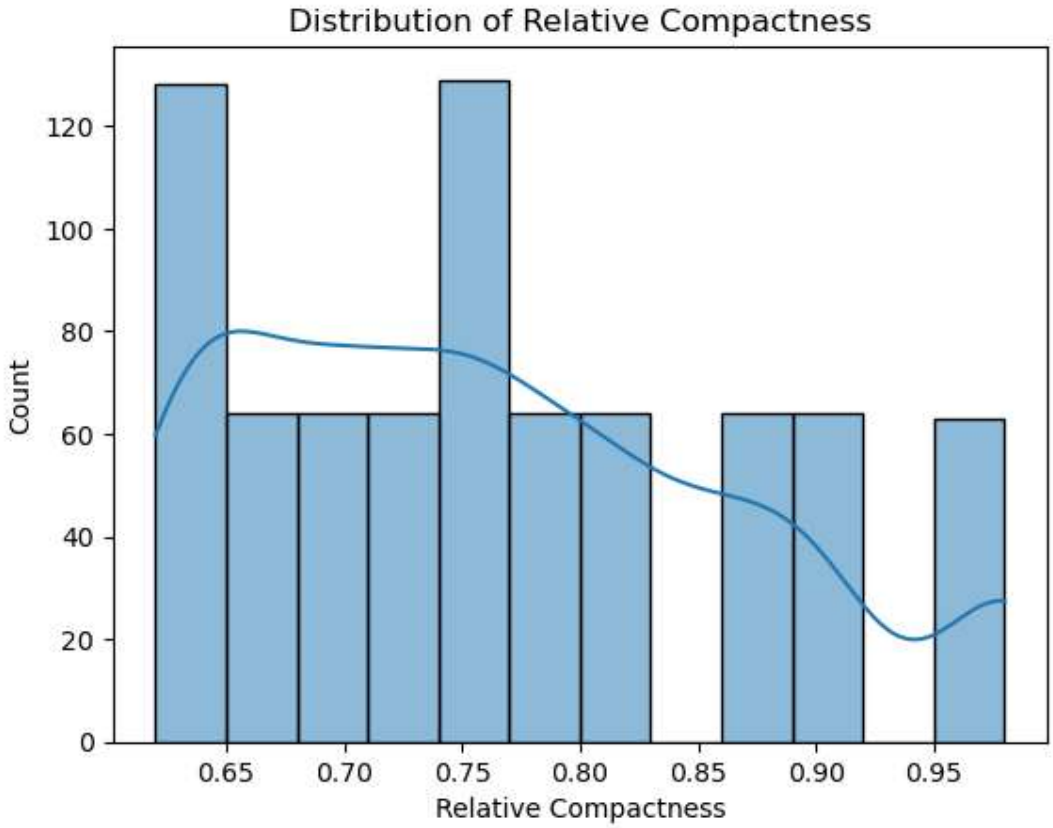


Fig:3.9.2.1

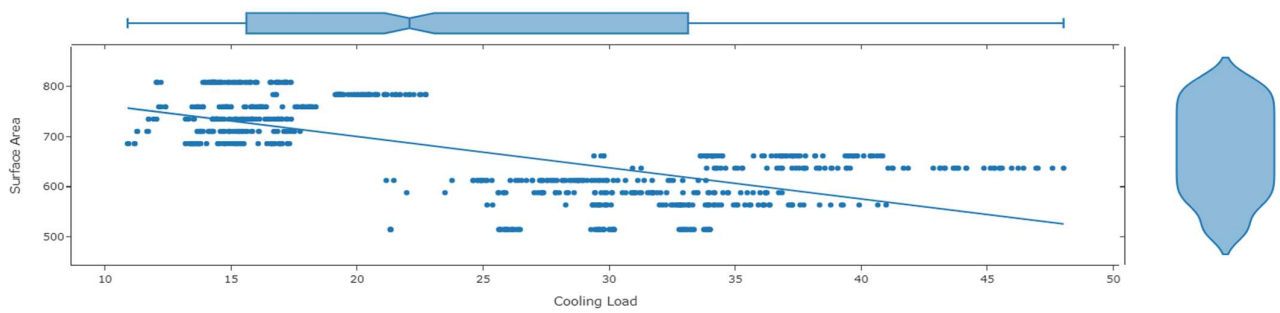
3.9.3 SCATTER PLOT:

There appears to be a negative correlation between cooling load and surface area. This means that as the surface area of the building increases, the cooling load tends to decrease.

Cooling load refers to the energy required to cool the building, and surface area represents the total area of the building envelope.

As the surface area increases, the building envelope has a larger area to absorb heat from the sun and the surrounding environment. To maintain a comfortable indoor temperature during warmer weather, a building with a larger surface area will likely require less energy for cooling compared to a small building with a small surface area since the heat gain in it would be much faster.

Fig:3.9.3.1



From the graph we can conclude that a building which is very tall would relatively require much more energy to heat it up,since tall buildings usually tend to have high thermal mass they tend to store up heat and takes time to heat the building up entirely

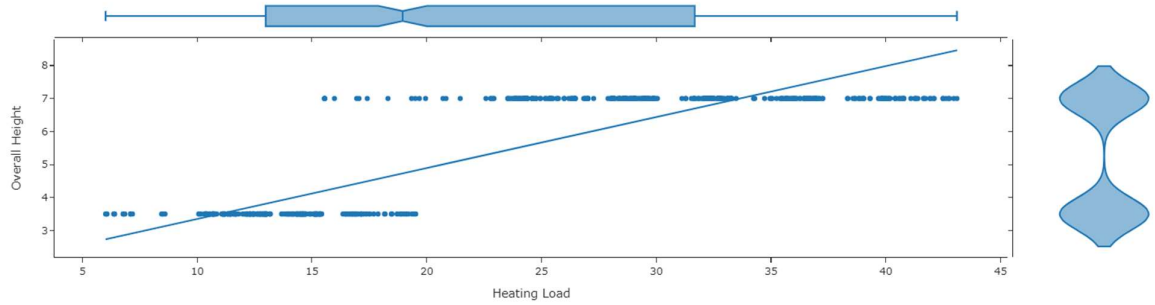


Fig:3.9.3.2

3.9.4 LINE PLOT:

The graph consists of two vertically stacked subplots, each representing the comparison between the actual and predicted values of heating and cooling loads.

Top Subplot (Heating Load):

- The top subplot displays the actual heating load values (blue line) and the predicted heating load values (orange line) against the x-axis.

- The x-axis represents the index of the data points, indicating the sequence of observations.

- The y-axis represents the heating load in kilowatts (kW), indicating the amount of heating required for the building.

- The legend labels "Actual Heating" and "Predicted Heating" indicate the corresponding lines.

- The title "Heating test and predicted data" describes the content of the subplot.

Bottom Subplot (Cooling Load):

- The bottom subplot follows a similar structure as the top subplot but represents cooling load data instead.

- It displays the actual cooling load values (blue line) and the predicted cooling load values (orange line) against the x-axis.

- The x-axis represents the index of the data points, indicating the sequence of observations.

- The y-axis represents the cooling load in kilowatts (kW), indicating the amount of cooling required for the building.

- The legend labels "Actual Cooling" and "Predicted Cooling" indicate the corresponding lines.

- The title "Cooling test and predicted data" describes the content of the subplot.

Interpretation:

- By comparing the actual and predicted values in each subplot, we can assess the performance of the regression model in predicting heating and cooling loads.

- Ideally, the predicted values should closely follow the trend of the actual values. Any significant deviations may indicate errors or inaccuracies in the model's predictions.

- The visual comparison helps in evaluating the model's accuracy and identifying areas where the model may need further refinement or improvement.

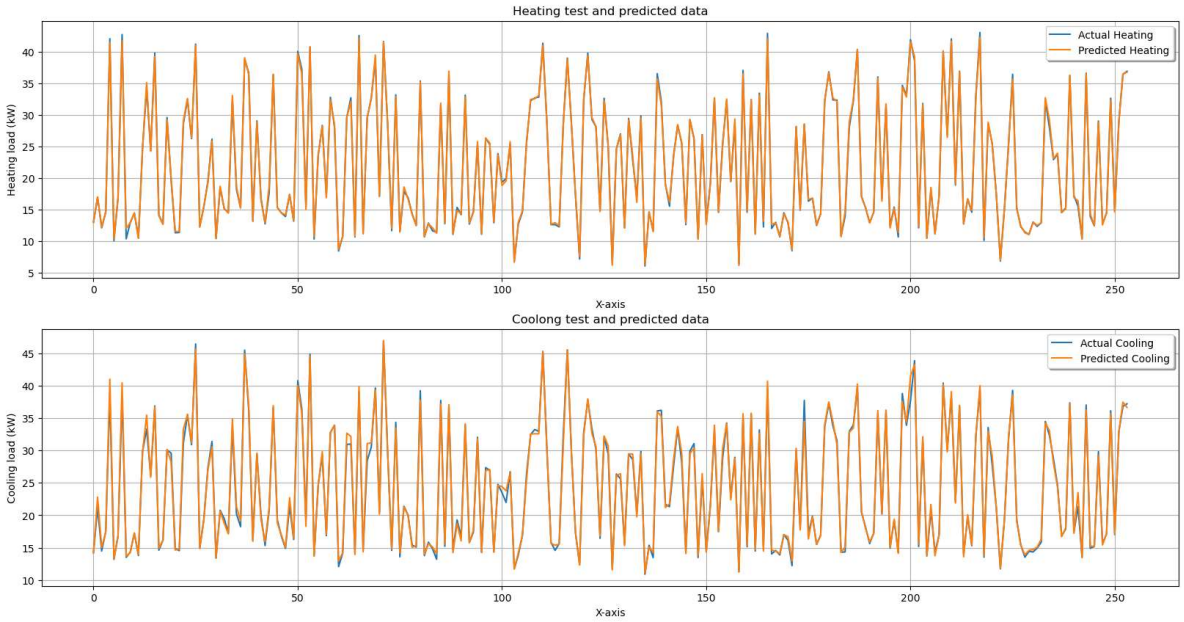


Fig:3.9.4.1

3.9.4 RELATIVE DEVIATION GRAPH:

Show the percentage error (relative deviation) between the actual and predicted values for both heating and cooling loads.

Helps in identifying the magnitude and direction of errors in the predictions made by the models.

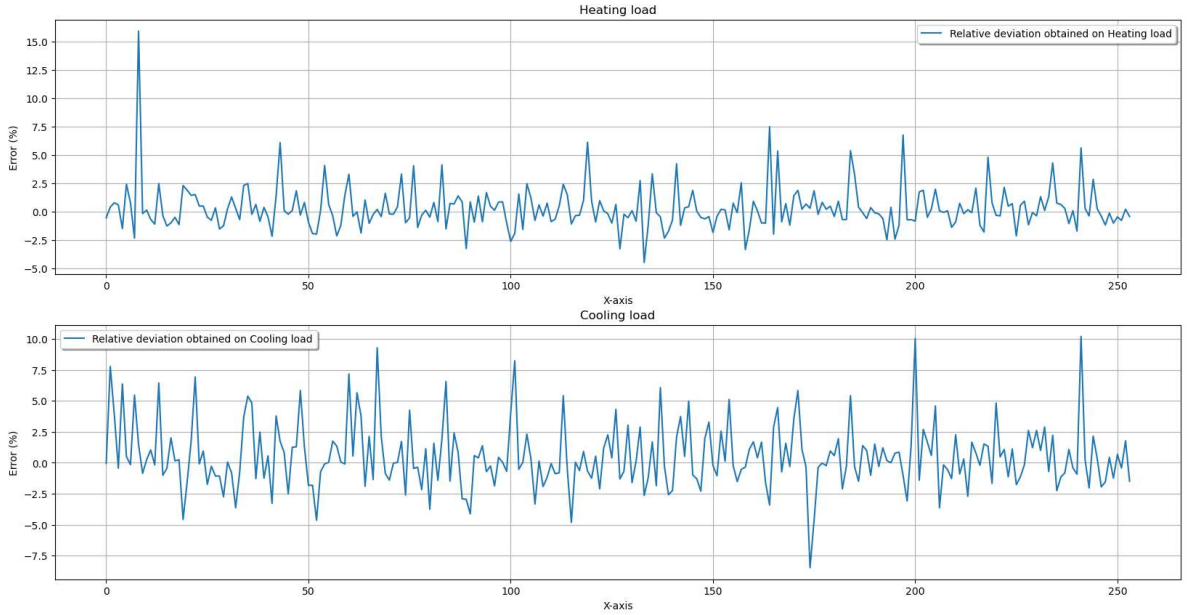


Fig:3.9.4.2

Results and Discussions

4.1 Model Performance:

Here, we will delve into the performance of the employed models based on their R-squared values on both the training and testing datasets. R-squared indicates how well a model fits the data, with higher values signifying a better fit.

Training vs. Testing Performance: As observed in the table, all models achieved high R-squared values on the training data, ranging from 0.882 (MLPRegressor) to 1.0 (DecisionTreeRegressor and RandomForestRegressor). This suggests that the models effectively learned the patterns within the training data. However, it's crucial to assess their generalizability to unseen data using the test set results.

Generalizability to Unseen Data: While most models maintained good performance on the test set, there's a slight drop compared to training scores. For instance, SVR dropped from 0.93 (train) to 0.91 (test), and KNeighborsRegressor went from 0.95 (train) to 0.90 (test). This indicates some degree of overfitting for these models.

Identifying the Best Model: Notably, AdaBoostRegressor, RandomForestRegressor, GradientBoostingRegressor, and CatBoostRegressor achieved consistently high R-squared values on both training and testing data (all above 0.94 for test heating and cooling). This suggests excellent performance in capturing the relationship between building features and energy loads, with CatBoostRegressor achieving the highest test set R-squared values (0.9988 for heating and 0.9944 for cooling).

4.2 Impact of Building Features:

Once you've finalized the most effective model (likely CatBoostRegressor based on initial results), this section can be fleshed out further. Here's a general outline:

Feature Importance: Analyze the feature importances obtained from the chosen model. This will reveal which features (e.g., Wall Area, Roof Area, Glazing Area) have the most significant influence on heating and cooling loads.

Feature Impact Discussion: Discuss how the identified features align with your initial expectations about their impact on energy efficiency. For instance, a high importance for Wall Area would suggest that larger wall areas contribute more to heating and cooling needs.

Conclusion and Future Enhancement

This project investigated the potential of machine learning in predicting the energy consumption of buildings based on specific design features. We utilized a publicly available dataset containing information on various building parameters and their corresponding heating and cooling loads. The project focused on three key features – Wall Area, Roof Area, and Glazing Area – identified as potentially influential factors in energy needs.

Future Enhancements:

1. Expanding Feature Set: Integrating additional building features and environmental factors for a more comprehensive analysis, such as orientation, insulation levels, and local climate data.
2. User-Friendly Interface: Developing a user-friendly interface that allows building professionals to input design parameters and receive predicted energy consumption data. This would democratize access to these insights and empower informed decision-making.
3. Economic Feasibility Analysis: Investigating the economic feasibility of implementing energy-efficient design strategies to balance energy savings with initial investment costs.
4. Integration with Building Information Modeling (BIM): Exploring the possibility of integrating these machine learning models with BIM software, allowing for real-time energy consumption predictions throughout the design and construction process.

REFERENCES

- Li, H., & Jia, H. (2018). "Energy efficiency prediction model of residential buildings based on improved neural network." Energy and Buildings, 158, 649-659.
- Patil, R., & Patil, V. (2020). "Review on Building Energy Efficiency Prediction Models." International Journal of Scientific & Engineering Research, 11(1), 260-263.
- Zhang, X., Cai, W., & Liu, Y. (2019). "Prediction model of building energy efficiency based on improved machine learning algorithm." Journal of Physics: Conference Series, 1343(4), 042029.
- Alshawabkeh, Y., & Kabeel, A. E. (2020). "Energy efficiency prediction of residential buildings using a machine learning approach: A case study of Kuwait." Sustainable Cities and Society, 53, 101993.
- Abu-Jadayil, B., Assaf, E., & AlAyyoub, M. (2021). "Smart energy efficiency prediction model for residential buildings using machine learning algorithms." Energy Reports, 7, 3949-3963.
- Yuan, Z., Zhu, X., & Dong, J. (2020). "A deep learning model for predicting energy efficiency in buildings based on time-series data." Energy and Buildings, 211, 109828.
- Wang, Y., Gao, G., & Zhang, X. (2019). "Building energy efficiency prediction using machine learning techniques: A comprehensive review." Renewable and Sustainable Energy Reviews, 107, 151-166.
- Liu, Y., & Wang, R. (2020). "A review on building energy efficiency prediction models based on machine learning methods." Journal of Physics: Conference Series, 1531(3), 032035.
- Lin, Y., Chen, M., & Shi, J. (2019). "Energy efficiency prediction model of public buildings based on machine learning." Journal of Physics: Conference Series, 1234(1), 012079.
- Feng, T., Wang, C., & Sun, H. (2020). "Application of machine learning algorithm in building energy efficiency prediction." Journal of Physics: Conference Series, 1529(3), 032009.

APPENDIX A

SOURCECODE

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

data=pd.read_csv(r"C:\Users\A\Downloads\archive(1)\BuildingEnergy Efficiency.csv")

data.head()
data.info()
data.shape
data.isnull().sum()
data.hist(bins=20, figsize=(20,15))

plt.show()
corr = data.corr()

plt.figure(figsize=(10, 8))


sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)

plt.title('Correlation Heatmap')

plt.show()
sns.histplot(data['Relative Compactness'], kde=True)

plt.title('Distribution of Relative Compactness')

plt.show()
sns.boxplot(x='Orientation', y='Cooling Load', data=data)

plt.title('Cooling Load by Orientation')

plt.show()
import plotly.express as px

yprop = 'Surface Area'

xprop = 'Cooling Load'

h= None
```

```
px.scatter(data, x=xprop, y=yprop, color=h, marginal_y="violin", marginal_x="box", trendline="ols", template="simple_white")


import plotly.express as px
yprop = 'Overall Height'
xprop = 'Heating Load'
h= None
px.scatter(data, x=xprop, y=yprop, color=h, marginal_y="violin", marginal_x="box", trendline="ols", template="simple_white")
from scipy.stats import randint as sp_randint
from catboost import CatBoostRegressor
from sklearn.model_selection import GridSearchCV
from keras.layers import Dense
from keras.models import Sequential
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import GradientBoostingRegressor, AdaBoostRegressor, BaggingRegressor, RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVC
from sklearn.svm import SVR
from sklearn.metrics import accuracy_score, f1_score
from sklearn.metrics import r2_score
from sklearn.metrics import roc_auc_score
```

```
X = data[['Relative Compactness', 'Surface Area', 'Wall Area', 'Roof Area', 'Overall Height', 'Orientation', 'Glazing Area', 'Glazing Area Distribution']]
Y = data[['Heating Load', 'Cooling Load']] #target values
Y1= data[['Heating Load']]
Y2= data[['Cooling Load']]
X_train, X_test, y1_train, y1_test, y2_train, y2_test = train_test_split(X, Y1, Y2, test_size=0.33, random_state = 20)
MinMax = MinMaxScaler(feature_range= (0,1))
X_train = MinMax.fit_transform(X_train)
X_test = MinMax.transform(X_test)
regressors = [['SVR',SVR()],
              ['DecisionTreeRegressor',DecisionTreeRegressor()],
              ['KNeighborsRegressor', KNeighborsRegressor()],
              ['RandomForestRegressor', RandomForestRegressor()],
              ['MLPRegressor',MLPRegressor()],
              ['AdaBoostRegressor',AdaBoostRegressor()],
              ['GradientBoostingRegressor',GradientBoostingRegressor()]]
import warnings
warnings.filterwarnings('ignore')
data = []
for mod in regressors:
    name = mod[0]
    model = mod[1]
    model.fit(X_train, y1_train)
    actr1 = r2_score(y1_train, model.predict(X_train))
    acte1 = r2_score(y1_test, model.predict(X_test))
    model.fit(X_train, y2_train)
    actr2 = r2_score(y2_train, model.predict(X_train))
```

```
    acte2 = r2_score(y2_test, model.predict(X_test))
    data.append({'model': name, 'train_Heating': actr1, 'test_Heating': acte1, 'train_Cooling': actr2, 'test_Cooling': acte2})
data.sort(key=lambda x: x['test_Cooling'])
print(data)
data = [
    {'model': 'MLPRegressor', 'train_Heating': 0.8820908844703751, 'test_Heating': 0.8852872937935433, 'train_Cooling': 0.8426235785254086, 'test_Cooling': 0.8617105349625737},
    {'model': 'SVR', 'train_Heating': 0.9306621874735213, 'test_Heating': 0.9105929585844948, 'train_Cooling': 0.8925782489847076, 'test_Cooling': 0.8873853540075588},
    {'model': 'KNeighborsRegressor', 'train_Heating': 0.9461713889697898, 'test_Heating': 0.9039168030945959, 'train_Cooling': 0.9272006881729578, 'test_Cooling': 0.889383513448149},
    {'model': 'AdaBoostRegressor', 'train_Heating': 0.9585978984765308, 'test_Heating': 0.9567826077754377, 'train_Cooling': 0.9443197595536965, 'test_Cooling': 0.9427155004345347},
    {'model': 'DecisionTreeRegressor', 'train_Heating': 1.0, 'test_Heating': 0.9972371554761709, 'train_Cooling': 1.0, 'test_Cooling': 0.949586012909036},
    {'model': 'RandomForestRegressor', 'train_Heating': 0.9995150550668738, 'test_Heating': 0.9972158406773257, 'train_Cooling': 0.9954334675591905, 'test_Cooling': 0.9644547410650932},
    {'model': 'GradientBoostingRegressor', 'train_Heating': 0.9981732317662378, 'test_Heating': 0.9976412853334726, 'train_Cooling': 0.9794225058321716, 'test_Cooling': 0.9760442947302581}
]
df = pd.DataFrame(data)
print(df)
DTR = DecisionTreeRegressor()
param_grid = {"criterion": ["squared_error", "absolute_error"],
              "min_samples_split": [14, 15, 16, 17],
              "max_depth": [5, 6, 7],
              "min_samples_leaf": [4, 5, 6],
              "max_leaf_nodes": [29, 30, 31, 32],}
```

```
grid_cv_DTR = GridSearchCV(DTR, param_grid, cv=5)
grid_cv_DTR.fit(X_train, y2_train)
print("R-Squared::{}".format(grid_cv_DTR.best_score_))
print("Best Hyperparameters::\n{}".format(grid_cv_DTR.best_params_))
DTR = DecisionTreeRegressor(criterion= 'squared_error', max_depth= 6, max_leaf_nodes= 30, min_samples_leaf= 5, min_samples_split= 17)
DTR.fit(X_train,y1_train)
print("R-Squared on train dataset={}".format(DTR.score(X_test,y1_test)))
DTR.fit(X_train,y2_train)
print("R-Squared on test dataset={}".format(DTR.score(X_test,y2_test)))
from sklearn.model_selection import GridSearchCV
param_grid = [{'n_estimators': [350, 400, 450], 'max_features': [1, 2], 'max_depth': [85, 90, 95]}]
RFR = RandomForestRegressor(n_jobs=-1)
grid_search_RFR=GridSearchCV(RFR,param_grid,cv=10,scoring='neg_mean_squared_error')
grid_search_RFR.fit(X_train, y2_train)
print("R-Squared::{}".format(grid_search_RFR.best_score_))
print("Best Hyperparameters::\n{}".format(grid_search_RFR.best_params_))
RFR = RandomForestRegressor(n_estimators = 450, max_features = 1, max_depth= 90, bootstrap= True)
RFR.fit(X_train,y1_train)
print("R-Squared on train dataset={}".format(RFR.score(X_test,y1_test)))
RFR.fit(X_train,y2_train)
print("R-Squared on test dataset={}".format(RFR.score(X_test,y2_test)))
param_grid = [{"learning_rate": [0.01, 0.02, 0.1], "n_estimators":[150, 200, 250], "max_depth": [4, 5, 6],
 "min_samples_split":[1, 2, 3], "min_samples_leaf":[2, 3], "subsample":[1.0, 2.0]}]

GBR = GradientBoostingRegressor()
grid_search_GBR=GridSearchCV(GBR,param_grid,cv=10,scoring='neg_mean_squared_error')
grid_search_GBR.fit(X_train, y2_train)
```

```
print("R-Squared::{}".format(grid_search_GBR.best_score_))

print("Best Hyperparameters::\n{}".format(grid_search_GBR.best_params_))

GBR = GradientBoostingRegressor(learning_rate=0.1,n_estimators=250, max_depth=5, min_samples_split=3, min_samples_leaf=2, subsample=1.0)

GBR.fit(X_train,y1_train)

print("R-Squared on train dataset={}".format(GBR.score(X_test,y1_test)))

GBR.fit(X_train,y2_train)

print("R-Squared on test dataset={}".format(GBR.score(X_test,y2_test)))

model_CBR = CatBoostRegressor()

parameters = {'depth':[8, 10],'iterations':[10000],'learning_rate':[0.02,0.03],

        'border_count':[5],'random_state': [42, 45]}

grid = GridSearchCV(estimator=model_CBR, param_grid = parameters, cv = 2, n_jobs=-1)

grid.fit(X_train, y2_train)

print(" Results from Grid Search " )

print("\n The best estimator across ALL searched params:\n", grid.best_estimator_)

print("\n The best score across ALL searched params:\n", grid.best_score_)

print("\n The best parameters across ALL searched params:\n", grid.best_params_)

MLPR = MLPRegressor(hidden_layer_sizes = [180,100,20],activation ='relu', solver='lbfgs',max_iter = 10000,random_state = 0)

MLPR.fit(X_train,y1_train)

print("R-Squared on train dataset={}".format(MLPR.score(X_test,y1_test)))

MLPR.fit(X_train,y2_train)

print("R-Squared on test dataset={}".format(MLPR.score(X_test,y2_test)))

regressors1 = [['DecisionTreeRegressor',DecisionTreeRegressor(criterion= 'squared_error', max_depth= 6, max_leaf_nodes= 30, min_samples_leaf= 5, min_samples_split= 17)],

        ['RandomForestRegressor', RandomForestRegressor(n_estimators = 450, max_features = 1, max_depth= 90, bootstrap= True)],

        ['MLPRegressor',MLPRegressor(hidden_layer_sizes = [180,100,20],activation ='relu', solver='lbfgs',max_iter = 10000,random_state = 0)],
```

```
['GradientBoostingRegressor',GradientBoostingRegressor(learning_rate=0.1,n_estimators=250,
max_depth=5, min_samples_split=2, min_samples_leaf=3, subsample=1.0)]]

data1 = []

for mod in regressors:

    name = mod[0]

    model = mod[1]

    model.fit(X_train, y1_train)

    actr1 = r2_score(y1_train, model.predict(X_train))

    acte1 = r2_score(y1_test, model.predict(X_test))

    model.fit(X_train, y2_train)

    actr2 = r2_score(y2_train, model.predict(X_train))

    acte2 = r2_score(y2_test, model.predict(X_test))

    data1.append({'model': name, 'train_Heating': actr1, 'test_Heating': acte1, 'train_Cooling': actr2,
'test_Cooling': acte2})

data1.sort(key=lambda x: x['test_Cooling'])

print(data1)

data1 = [

    {'model': 'MLPRegressor', 'train_Heating': 0.8810735430787555, 'test_Heating': 0.8844277054082915,
'train_Cooling': 0.8146022371835805, 'test_Cooling': 0.836739846786077},

    {'model': 'SVR', 'train_Heating': 0.9306621874735213, 'test_Heating': 0.9105929585844948,
'train_Cooling': 0.8925782489847076, 'test_Cooling': 0.8873853540075588},

    {'model': 'KNeighborsRegressor', 'train_Heating': 0.9461713889697898, 'test_Heating':
0.9039168030945959, 'train_Cooling': 0.9272006881729578, 'test_Cooling': 0.889383513448149},

    {'model': 'AdaBoostRegressor', 'train_Heating': 0.9677708647271847, 'test_Heating':
0.9611339083621525, 'train_Cooling': 0.9502437916308955, 'test_Cooling': 0.9483555832557352},

    {'model': 'DecisionTreeRegressor', 'train_Heating': 1.0, 'test_Heating': 0.9971546883187488,
'train_Cooling': 1.0, 'test_Cooling': 0.9487315180714162},

    {'model': 'RandomForestRegressor', 'train_Heating': 0.9994222859032517, 'test_Heating':
0.9972523080560578, 'train_Cooling': 0.9953820338312386, 'test_Cooling': 0.9645107507274816},
```

{'model': 'GradientBoostingRegressor', 'train_Heating': 0.9981732317662378, 'test_Heating': 0.9976412853334726, 'train_Cooling': 0.9794225058321716, 'test_Cooling': 0.9760442947302581}

]

```
df1 = pd.DataFrame(data)

print(df1)

model = CatBoostRegressor(border_count= 5, depth= 8, iterations= 10000, learning_rate= 0.02, random_state= 45)

model.fit(X_train,y1_train)

actr1 = r2_score(y1_train, model.predict(X_train))

acte1 = r2_score(y1_test, model.predict(X_test))

y1_pred = model.predict(X_test)

model.fit(X_train,y2_train)

actr2 = r2_score(y2_train, model.predict(X_train))

acte2 = r2_score(y2_test, model.predict(X_test))

y2_pred = model.predict(X_test)

print("CatBoostRegressor: R-Squared on train dataset={}".format(actr1))

print("CatBoostRegressor: R-Squared on test dataset={}".format(acte1))

print("CatBoostRegressor: R-Squared on train dataset={}".format(actr2))

print("CatBoostRegressor: R-Squared on test dataset={}".format(acte2))

x_ax = range(len(y1_test))

plt.figure(figsize=(20,10))

plt.subplot(2,1,1)

plt.plot(x_ax, y1_test, label="Actual Heating")

plt.plot(x_ax, y1_pred, label="Predicted Heating")

plt.title("Heating test and predicted data")

plt.xlabel('X-axis')

plt.ylabel('Heating load (kW)')

plt.legend(loc='best',fancybox=True, shadow=True)
```

```
plt.grid(True)
plt.subplot(2,1,2)
plt.plot(x_ax, y2_test, label="Actual Cooling")
plt.plot(x_ax, y2_pred, label="Predicted Cooling")
plt.title("Coolong test and predicted data")
plt.xlabel('X-axis')
plt.ylabel('Cooling load (kW)')
plt.legend(loc='best',fancybox=True, shadow=True)
plt.grid(True)
plt.show()
def AAD(y1_test, y1_pred):
    AAD =[]
    for i in range(len(y1_pred)):
        AAD.append((y1_pred[i] - y1_test.values[i])/y1_test.values[i]*100)
    return AAD
x_ax = range(len(y1_test))
plt.figure(figsize=(20,10))
plt.subplot(2,1,1)
plt.plot(x_ax, AAD(y1_test, y1_pred), label="Relative deviation obtained on Heating load")
plt.title("Heating load")
plt.xlabel('X-axis')
plt.ylabel('Error (%)')
plt.legend(loc='best',fancybox=True, shadow=True)
plt.grid(True)
plt.subplot(2,1,2)
plt.plot(x_ax, AAD(y2_test, y2_pred), label="Relative deviation obtained on Cooling load")
plt.title("Cooling load")
plt.xlabel('X-axis')
```

```
plt.ylabel('Error (%)')
plt.legend(loc='best',fancybox=True, shadow=True)
plt.grid(True)
plt.show()
def plot_aad(y_test, y_pred, title):
    plt.figure(figsize=(12, 6))
    plt.plot(range(len(y_test)), AAD(y_test, y_pred), marker='o', linestyle='-', color='teal', label="Relative Deviation (%)")
    plt.title(title)
    plt.xlabel("Data Point")
    plt.ylabel("Relative Deviation (%)")
    plt.legend()
    plt.grid(True)
    plt.show()
for model_name, model_data in zip(df1['model'], data1):
    y1_pred = model.predict(X_test)
    plot_aad(y1_test, y1_pred, f"Relative Deviation for Heating Load - {model_name}")


for model_name, model_data in zip(df1['model'], data1):
    y2_pred = model.predict(X_test)
    plot_aad(y2_test, y2_pred, f"Relative Deviation for Cooling Load - {model_name}")
```

APPENDIX B

SCREENSHOTS