Energy Efficiency Foretold: Optimizing Predictive Accuracy in Building Performance with Machine Learning

Ze Liu, Weijie Chen
[1] Department of Mechatronic Systems Engineering, Simon Fraser University,
13450 102 Ave 250, Surrey, BC V3T 0A3, Canada

April 15th, 2024

1 Introduction

In the context of growing environmental concerns and the increasing need for sustainable building practices, the ability to accurately assess a building's energy efficiency has become crucial. The Building Energy Star Score, a key metric influencing both environmental sustainability and operational costs, serves as the focal point for this research. Utilizing the expansive NYC benchmarking dataset hosted on Kaggle, which includes energy consumption data across over 11,000 buildings and 60 variables, this study aims to leverage machine learning algorithms to predict the Energy Star Score more effectively.

Historically, Gradient Boosting has been extensively used in various domains such as finance, healthcare, and social media analytics due to its ability to handle diverse datasets and complex predictive problems. However, its limitations, including susceptibility to overfitting and noise, necessitate exploring more robust alternatives. In this pursuit, the Random Forest algorithm, known for its ensemble approach using multiple decision trees to improve prediction accuracy and robustness, is investigated as a primary analytical tool.

This research is inspired by a prior machine learning project walkthrough that utilized Gradient Boosting, documented in a GitHub repository. The current study adapts this approach by implementing the Random Forest algorithm to determine its efficacy under similar conditions, focusing on:

- Comparing the accuracy and efficiency of Random Forest with Gradient Boosting in predicting the Energy Star Score.
- Evaluating the implications of utilizing Random Forest concerning model interpretability and its capability to handle overfitting.

The objectives of this investigation are to identify key predictors of the Energy Star Score, develop and assess both regression and classification models, and provide insights that could influence energy usage patterns and strategies for enhancing building efficiencies. By addressing these aspects, this study not only seeks to validate the effectiveness of the Random Forest algorithm in scenarios traditionally dominated by Gradient Boosting but also aims to enrich the discourse on sustainable building management through advanced machine learning methodologies. This research promises to extend the utility of machine learning in environmental science, offering scalable models adaptable to various urban contexts and building types.

2 Methods

2.1 Data Preprocessing

Data preprocessing is a critical step to ensure the quality and usability of the data for modeling. The process in this project includes several key stages:

2.1.1 Data Cleaning and Formatting

Initially, the dataset contains entries marked as 'Not Available', which are not suitable for analysis. These entries are replaced with NaN (Not a Number) to standardize the missing value format across the dataset. This is achieved using the following Python code:

```
data = data.replace({'Not Available': np.nan
    })
```

2.1.2 Type Conversion

To facilitate numerical analysis, it is essential that all data expected to be in numeric form is converted accordingly. This involves identifying columns that represent measurements such as square footage, energy usage (kBtu, kWh), emissions (Metric Tons CO2e), water usage (gallons), and scores. These columns are then explicitly converted to float type to ensure their numerical nature is preserved, as shown in the code snippet:

```
for col in data.columns:
    if any(metric in col for metric in ['ft2', 'kBtu', 'Metric Tons CO2e', 'kWh', 'therms', 'gal', 'Score']):
        data[col] = data[col].astype(float)
```

2.1.3 Handling Missing Data

An initial assessment of missing data is conducted to identify columns with substantial missing values. Columns where over 50% of the data is missing are considered for removal, as retaining them could compromise the integrity and accuracy of the modeling process. This decision is based on a threshold that balances data retention with quality.

2.1.4 Column Removal

Based on the missing data analysis, columns with missing data exceeding the 50% threshold are removed from the dataset. This step simplifies the dataset and focuses the analysis on the most reliable and informative attributes. For example, 11 columns with high levels of missing data were removed, reducing the total number of columns in the dataset and ensuring a cleaner dataset for modeling.

2.2 Data Visualization and Exploration

After the initial data cleaning, the next step involves detailed data visualization and further exploration to understand the distributions, identify any underlying patterns or anomalies, and prepare for more sophisticated cleaning and feature engineering. This stage includes creating visual plots to examine the relationships between features, the distribution of key variables, and further identification of outliers or unusual data points that might require additional preprocessing.

2.2.1 Initial Data Visualization

The analysis began with the exploration of the distribution of the Energy Star Scores across the dataset. A significant finding was the unusually high frequency of scores at the extreme values of 1 and 100, as shown in the following graph:
From the plot, we can observe that the distribution is peculiar, with scores of 1 and 100 comprising a very high proportion. This prompted further investigation into the nature of the scoring system.

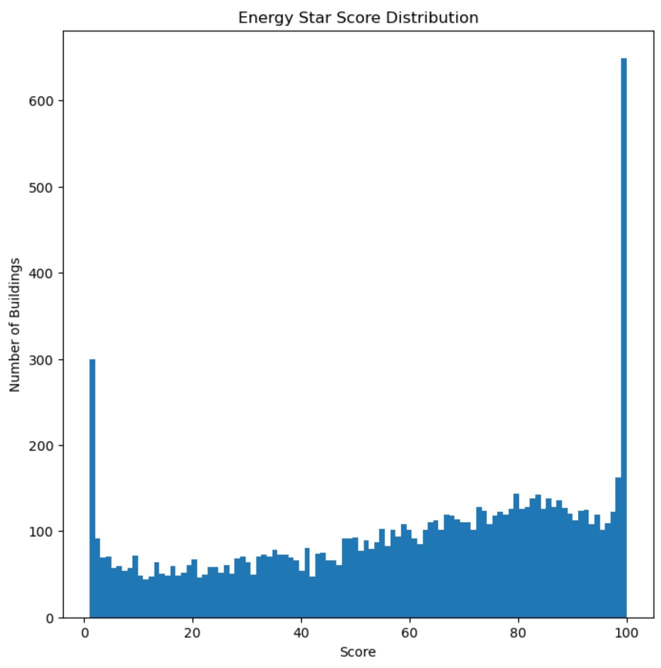


Figure 1: Distribution of Energy Star Scores

2.2.2 Investigating the Energy Star Score Reporting

With further research and data exploration, it was discovered that the Energy Star Score is reported by the building owners themselves. This self-reporting mechanism may lead some owners to report scores that reflect an idealized or incorrect understanding of the scoring scale, rather than being based on actual energy usage. Interestingly, some owners might mistakenly believe that a score of '1' represents the highest level of energy efficiency, which is not the case.

2.2.3 Rationale for Using EUI

Given the inconsistencies and potential misreporting in the Energy Star Score, the decision was made to use Energy Use Intensity (EUI) as a more reliable metric for analysis. EUI, calculated as the total energy used divided by the building's square footage, provides a direct measure of energy efficiency independent of subjective reporting. This metric is illustrated in the histogram below:

2.2.4 Descriptive Statistics and Outlier Analysis

The descriptive statistics of the Site EUI revealed an extremely skewed distribution, as indicated by the following metrics:

- Count: 11,583
- Mean: 280.071 kBtu/ft²
- Standard Deviation: 8,607.179 kBtu/ft²

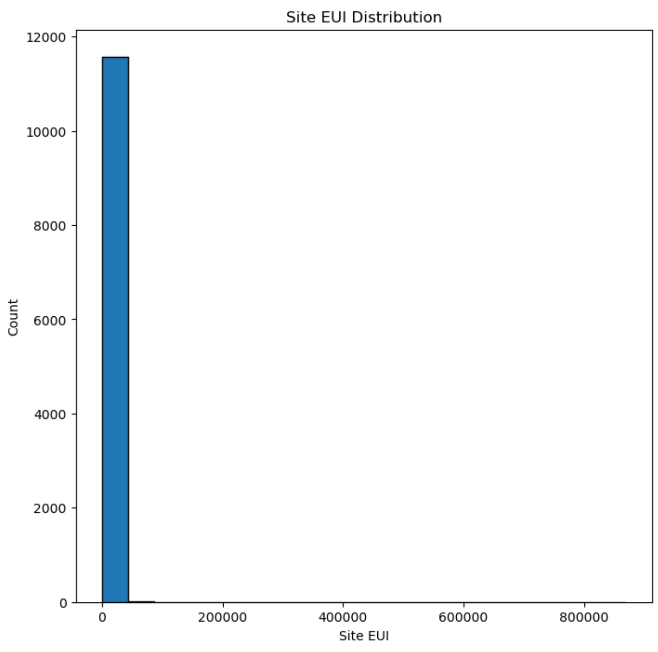


Figure 2: Histogram of Site EUI

- Minimum: 0.0 kBtu/ft²
- 25% (Q1): 61.8 kBtu/ft²
- 50% (Median): 78.5 kBtu/ft²
- 75% (Q3): 97.6 kBtu/ft²
- Maximum: 869,265.0 kBtu/ft²

The vast range between the minimum and maximum values, along with a high standard deviation, suggested the presence of significant outliers. A boxplot was used to visualize these outliers:

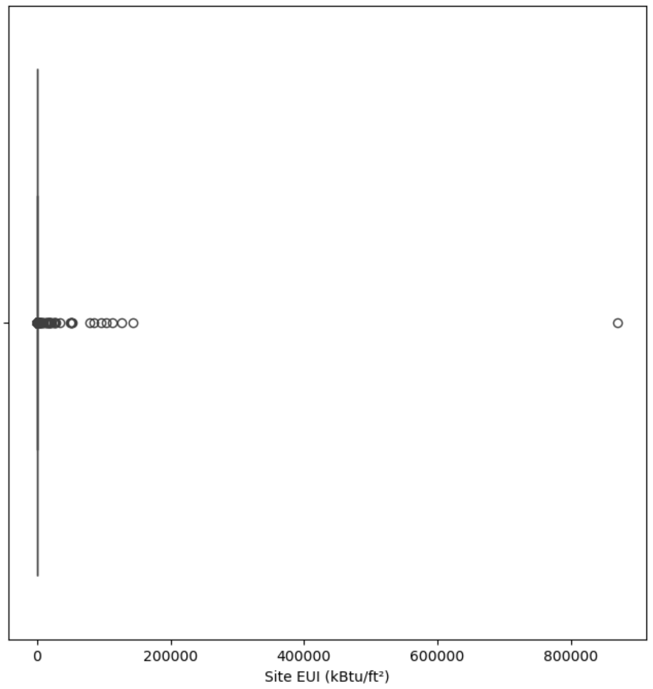


Figure 3: Boxplot before outlier removal

2.3 Outlier Removal

To address the issue of outliers, which could potentially skew the model's predictions, a method based on the interquartile range (IQR) was employed. Outliers were defined as observations that fall below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR. This approach aimed to trim the data to a more manageable and statistically representative dataset.

The adjusted boxplot, post-outlier removal, showed a much-improved distribution:

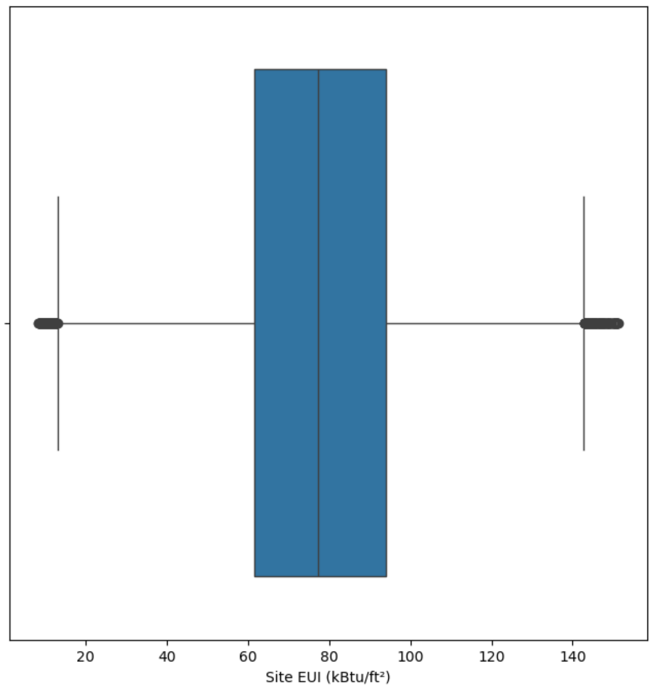


Figure 4: Boxplot after outlier removal

2.3.1 Refined Histogram Analysis

With outliers removed, a second histogram of Site EUI was plotted to assess the effect of this cleaning step on the data distribution:

The new histogram displayed a distribution that was closer to normal but still showed a positive skew. This normalization of data distribution is crucial for the effectiveness of many statistical modeling techniques, including regression analysis, which assumes normally distributed residuals.

The initial data visualization and cleaning stages were crucial in identifying and correcting for outliers, which enhances the reliability of subsequent modeling steps. The predictive model is likely to be more accurate and generalizable by adjusting the dataset to reflect the typical energy use patterns. The findings also highlight the importance of using objective metrics like EUI for building energy efficiency studies, avoiding reliance on potentially biased self-reported data.

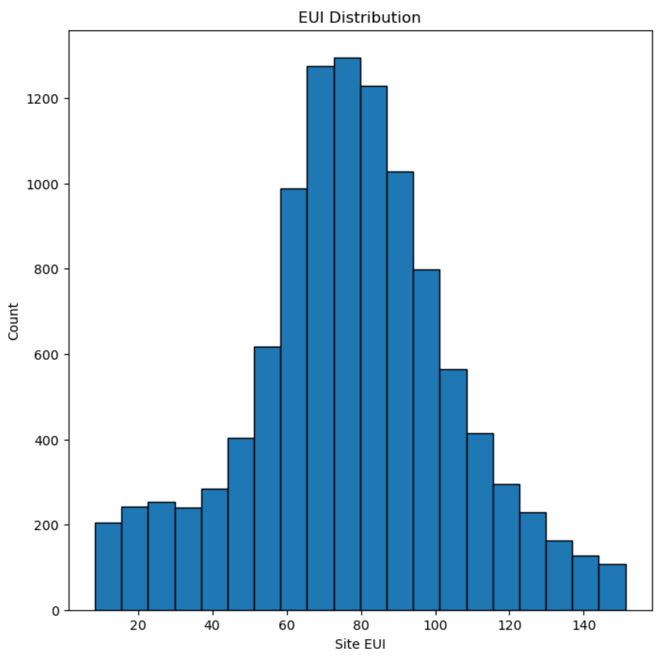


Figure 5: Histogram after outlier removal

2.3.2 Data Relationship Exploration

Analyzing Building Types

To understand the impact of categorical variables such as building type on the Energy Star Score, we employed density plots for visualization. Density plots are especially useful in this context because they provide a smoothed continuous estimate of the distribution of a variable. This makes them ideal for comparing the distribution of scores across different categories without the distractions and potential misinterpretations that can arise with histograms, which depend heavily on bin size.

Given the extensive diversity of building types in our dataset, to maintain clarity and focus in our visual analysis, we limited our density plots to building types represented by more than 100 observations. This threshold ensures that our visualizations are statistically meaningful and not skewed by outliers or sparse data. From the density plot, it is evident that different building types exhibit varying impacts on the Energy Star Score. This suggests that building type is a significant factor in determining energy efficiency scores, which underscores the necessity of treating building type as a separate variable in our analysis.

Exploring Borough Influence

Subsequently, we extended our analysis to include another categorical variable, the borough in which the building is located. A similar approach using density plots was applied to examine if the geographical location within the city influences the Energy Star Score in a manner akin to building type.

The analysis of the borough data reveals that the

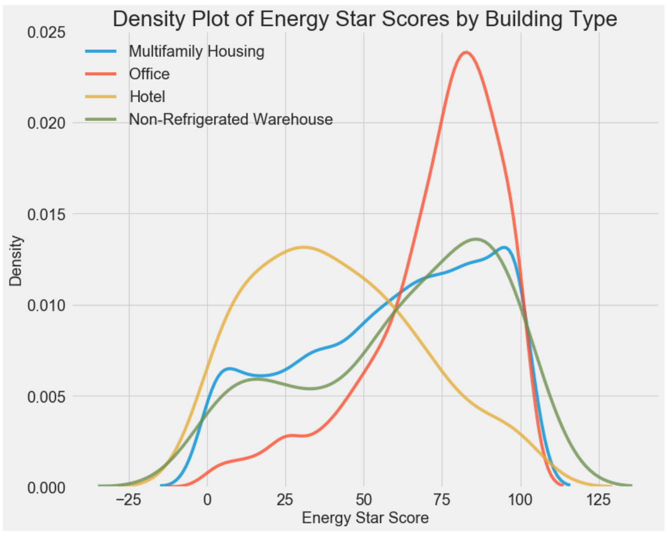


Figure 6: Density plot for various building types

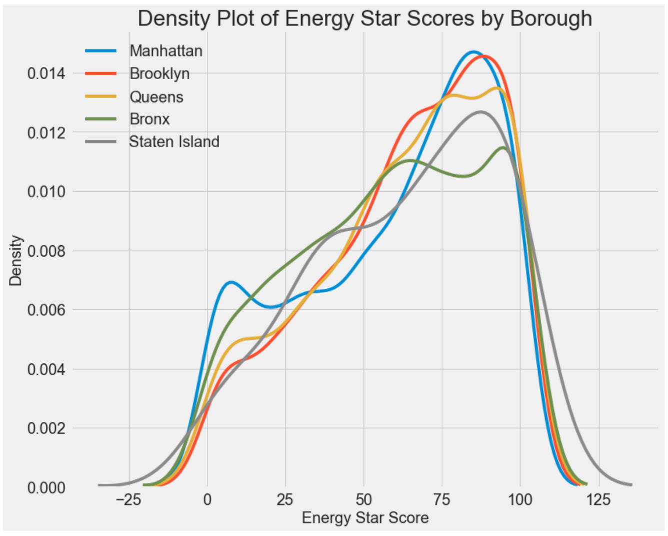


Figure 7: Density plot for Energy Star Scores across different boroughs

influence of the borough on the Energy Star Score distribution is less pronounced compared to that of building type. Although differences are visible, they are not as distinct. However, considering the potential for regional variations in building regulations, energy supply, and environmental factors, it may still be prudent to include borough as a categorical variable in the predictive model. This inclusion could help in refining the model's accuracy by accounting for subtle regional differences that might affect energy consumption and efficiency.

The exploration of categorical variables through density plots not only clarifies their respective influences on the Energy Star Score but also aids in making informed decisions about variable inclusion in the model. Building type has emerged as a crucial variable and will be treated with particular emphasis in the predictive modeling. Although borough shows a less significant impact, incorporating it as a categorical variable may enhance the model's comprehensiveness and its ability to generalize across different urban contexts.

By meticulously analyzing these categorical factors, we ensure that our model captures a broad spectrum of influences, thereby increasing the reliability and applicability of its predictions.

2.4 Exploration of Feature Correlations

Understanding Pearson Correlation Coefficient

To explore the relationships between features and the target variable (Energy Star Score), we utilized the Pearson Correlation Coefficient. This statistical measure helps identify the strength and direction of a linear relationship between two continuous variables. It is important to note that Pearson's coefficient only captures linear correlations and does not reflect non-linear relationships or interactions between features. However, it provides a valuable initial assessment for feature selection and trend identification.

Initial Correlation Analysis

Our initial correlation analysis revealed several strong negative relationships, particularly involving various forms of Energy Use Intensity (EUI). The most negative correlations with the score were seen with:

- Site EUI (kBtu/ft²): -0.723864
- Weather Normalized Site EUI (kBtu/ft²): -0.713993
- Weather Normalized Source EUI (kBtu/ft²): -0.645542
- Source EUI (kBtu/ft²): -0.641037

These results are intuitive; as EUI values increase, indicating higher energy consumption per square foot, the Energy Star Score typically decreases, reflecting poorer energy efficiency.

Addressing Non-linear Relationships

Given the limitations of Pearson's correlation in capturing non-linear relationships, we proceeded to transform the features using square root and natural log transformations, recalculating the correlations to potentially unveil more insights. We also one-hot encoded categorical variables like building type and borough to analyze their impact.

Enhanced Correlation Findings

After applying transformations and one-hot encoding, our correlation analysis provided the following insights:

- Most negative correlations remained with EUI metrics, even after transformations:
  - $\sqrt{\text{Site EUI (kBtu/ft2)}}$: -0.699817
  - $\log(\text{Weather Normalized Site EUI (kBtu/ft2)})$: -0.601332
- The most positive correlations observed were:
  - Largest Property Use Type_Office: 0.158484
  - Borough_Brooklyn: 0.050486

These findings underscore the significance of EUI in predicting the Energy Star Score. The transformations did not fundamentally change the relationships, confirming the robustness of EUI as a predictor.

Visualizing Significant Correlations

To better understand these relationships, we will now graph the most significant correlation, Site EUI (kBtu/ft²), and use building type as a categorical variable to color the graph, which can illustrate how different building types might affect this relationship.

Two-Variable Plots for Deeper Insights

In visualizing the relationship between two variables, scatterplots are invaluable. They not only show the correlation but also allow for the inclusion of a third variable through visual means such as the color or size of the markers.

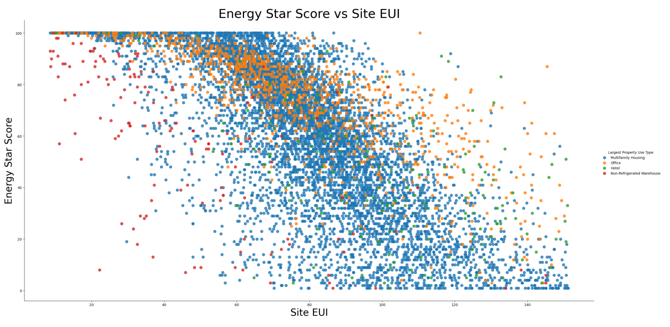


Figure 8: Energy Star Score vs. Site EUI Scatter Plot, Blue Multifamily Housing, Yellow Office, Green Hotel, Red Non-Regirgerated Warehouse

This scatter plot clearly demonstrates the negative relationship between Site EUI and the Energy Star Score. Although the relationship shows a correlation coefficient of -0.7, indicating a strong negative trend, it is not perfectly linear. This feature, however, appears to be crucial for predicting a building's score, and variations by building type (as indicated by marker color) can provide additional contextual insights into how different types of buildings consume energy.

By exploring these correlations and visualizing them, we can gain a deeper understanding of the factors that influence building energy efficiency, guiding feature selection and model development to predict Energy Star Scores more effectively.

2.5 Utilizing Pair Plots for Comprehensive Analysis

As we conclude our exploratory data analysis, utilizing a Pair Plot provides an extensive overview by showing relationships between multiple variable pairs simultaneously. A Pair Plot is particularly useful in a multivariable dataset as it includes scatterplots for each pair of variables and histograms for individual variables on the diagonal. This comprehensive visualization helps in identifying patterns, correlations, and potential outliers across different dimensions of the dataset.

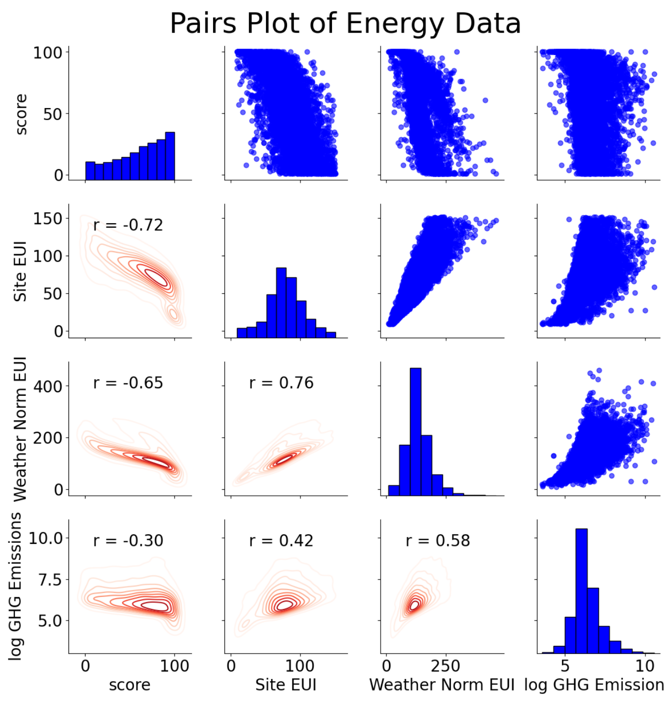


Figure 9: Example of a Pair Plot showing multiple variable relationships

Interpreting the Pair Plot

To effectively interpret the Pair Plot,

1. **Diagonal Histograms**: Each diagonal plot in the matrix represents the distribution of a single variable. Analyzing these can help identify skewness, peaks, and the spread of the data, which are critical for understanding variable characteristics.

2. **Off-diagonal Scatterplots**: Each off-diagonal plot shows the relationship between two variables. For example, to assess the relationship between the Energy Star Score and the logarithm of Greenhouse Gas (GHG) Emissions, locate the 'score' column and the 'log GHG Emissions' row. The plot at their intersection reveals the nature of their relationship.

   - **Interpreting Correlations**: In the specified example, the plot shows a correlation coefficient of -0.35, indicating a moderate negative relationship between the score and log GHG Emissions. This suggests that as GHG emissions increase, the Energy Star Score tends to decrease, highlighting the impact of emissions on building efficiency scores.
   - At the intersection of 'score' and 'log GHG Emissions', the scatterplot visualizes this negative correlation. Observing the spread and grouping of data points can further elucidate the strength and consistency of this relationship.

2.6 Introduction to Collinearity

Collinear features in a dataset are variables that are highly correlated with one another. They can adversely affect the performance of machine learning models by causing high variance in the estimation of regression coefficients, which can lead to less reliable predictions. Collinearity often complicates the interpretation of model outputs because it becomes challenging to discern the effect of each individual feature.

Identifying Collinear Features

In our analysis, we identified pairs of features with strong correlations, indicating potential collinearity. For example, 'Weather Normalized Site EUI (kBtu/ft^2)' and 'Site EUI (kBtu/ft^2)' exhibit very similar behavior as demonstrated in the correlation analysis.

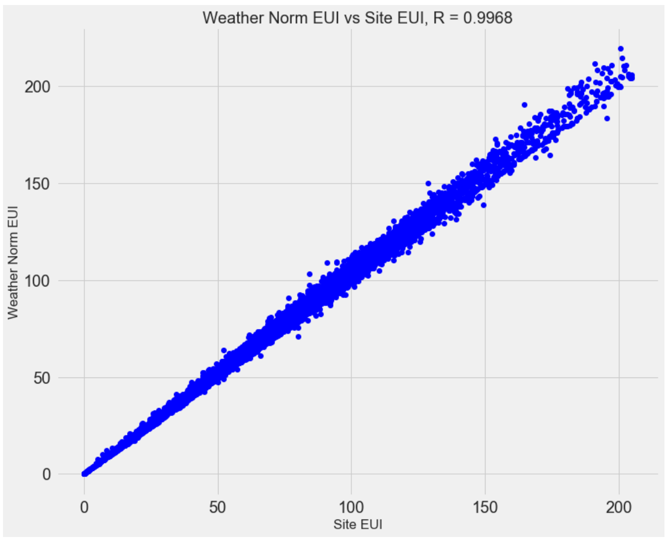


Figure 10: Correlation heatmap showing high correlation between Weather Normalized EUI and Site EUI

Approach to Handling Collinear Features

For our project, we chose to remove features that have a correlation coefficient greater than 0.6 with other features, considering them as collinear. This

decision is guided by the goal to simplify our model while maintaining robustness. After the removal of these collinear features, the shape of our feature matrix reduced from (11319, 110) to (11319, 65), illustrating a significant reduction in feature count.
While there are numerous methods for feature selection and dimensionality reduction, such as Principal Components Analysis (PCA) and Independent Components Analysis (ICA), these techniques were not employed in our project for specific reasons:

- **PCA** involves transforming the original variables into a new set of variables (principal components), which are linear combinations of the original variables. These components are chosen to capture the maximum variance within the data. However, one major drawback is that the principal components do not represent any physically meaningful quantities, making the interpretation of model results challenging.
- **ICA** is similar to PCA but focuses on maximizing the statistical independence of the estimated components, rather than the variance. This method is powerful for discovering underlying factors or sources in the data but, like PCA, results in components that may be difficult to interpret in a practical context.

Both PCA and ICA transform the features into new metrics that, while useful for some applications, render the model's decisions opaque. This lack of interpretability can be problematic in projects where understanding the influence of specific variables is crucial.

2.7 Data Preparation

Before proceeding with the split into training and testing sets, an important data cleaning step was to ensure the dataset's integrity by addressing infinite values. We replaced all instances of inf and -inf with NaN (Not a Number), which are placeholders for undefined or unrepresentable values. This step is crucial as it allows for proper handling during later stages of data imputation and model training.

```
data.replace([np.inf, -np.inf], np.nan,
    inplace=True)
```

2.7.1 Splitting the Data

The dataset was then split into training and testing sets. The division was made such that 30% of the data was reserved for testing, which is a common practice to ensure that the model can generalize well to new, unseen data. This split resulted in a training set comprising 6,622 examples, and a testing set comprising 2,839 examples. This partitioning ensures that both sets are large enough to represent the dataset's overall characteristics while allowing for comprehensive testing of the model's performance.

2.7.2 Establishing a Naive Baseline

To set a naive baseline, we used the median value of the target variable, as recommended by the machine learning course on Coursera Single Number Evaluation Metric. The rationale behind using the median as a baseline is that it is a robust measure of central tendency, not heavily influenced by outliers, making it a good conservative guess in the absence of any other information.
For our dataset, the baseline guess was determined to be a score of 66.00. This score represents the central tendency of the dataset's Energy Star Scores and provides a simple yet effective benchmark against which to measure the performance of our more complex predictive models.

2.7.3 Baseline Performance Evaluation

To evaluate the performance of our naive baseline, we utilized the Mean Absolute Error (MAE) as our loss function. MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's particularly useful because it quantifies the accuracy of a model into a single metric, providing a straightforward interpretation of model error.

```
baseline_mae = mean_absolute_error(
    test_labels, [66.00] * len(test_labels))
print(f"Baseline Performance on the test set:
     MAE = {baseline_mae:.4f}")
```

The calculated MAE for the baseline on the test set was 24.5164. This value establishes an initial benchmark for assessing the effectiveness of any subsequent models we develop. It sets a quantifiable goal: any predictive model that results in a lower MAE than this baseline would be considered an improvement, reflecting better accuracy in predicting the Energy Star Score.

2.7.4 Handling Missing Data

In our preprocessing workflow, the initial step was to address the missing data in our dataset. Choosing an imputation method can significantly influence the performance of subsequent models. For simplicity and effectiveness, we replaced missing values with the median of each column. Using the median is beneficial as it is less sensitive to outliers in the data compared to the mean, providing a more robust central value for each feature.

2.7.5 Data Scaling Using MinMaxScaler

Following imputation, we applied normalization to scale the feature data. Specifically, we used the `MinMaxScaler`, which adjusts the data to have a minimum of 0 and a maximum of 1. This scaling is particularly important because it ensures that all features contribute equally to the model's predictions, preventing features with larger scales from disproportionately influencing the model. Normalization is crucial for models like K-Nearest Neighbors and Support Vector Machines, which are sensitive to the scale of the input data.

2.8 Model Performance Comparison

With our data prepared, we proceeded to train several different models to identify which one best predicts the Energy Star Scores:

- **Linear Regression**: MAE = 12.6928
- **Support Vector Machine (SVM)** with C = 1000 and gamma = 0.1: MAE = 10.2213
- **Random Forest Regressor**: MAE = 9.4664
- **K-Neighbors Regressor** with n_neighbors=10: MAE = 11.9303

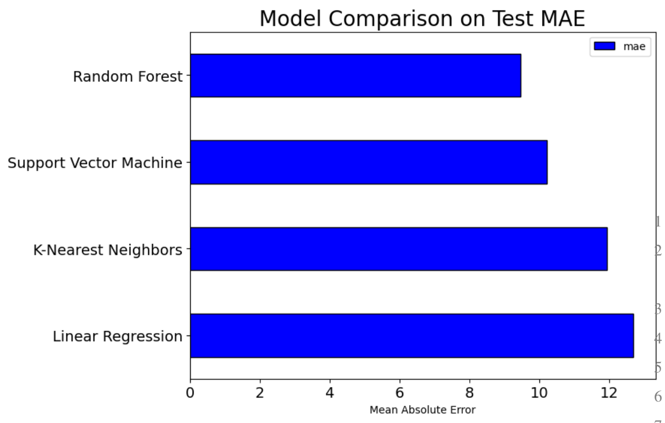


Figure 11: Performance comparison of different models on Energy Star Score prediction

Among these models, the Random Forest Regressor yielded the lowest Mean Absolute Error (MAE), indicating its superior performance in capturing the complexities of our dataset.

2.9 Choosing Random Forest for Hyperparameter Tuning

Based on the comparative analysis, we decided to focus on the Random Forest model for further optimization through hyperparameter tuning. This decision was influenced by several factors:

1. **Performance**: Random Forest had the best initial performance among the models tested, suggesting a good starting point for further refinement.
2. **Robustness**: Random Forest models are known for their robustness to outliers and variability in the data, which is particularly advantageous given the diverse range of buildings and their characteristics in our dataset.
3. **Feature Handling**: Random Forest can handle a large number of features and can provide insights into feature importance, which can be invaluable for understanding the factors influencing building energy efficiency.

By tuning the hyperparameters of the Random Forest model, such as the number of trees, depth of trees, and splitting criteria, we aim to enhance the model's accuracy and reduce overfitting, thereby achieving even lower MAEs.

2.10 Hyperparameter Tuning using Cross-Validation

The goal of hyperparameter tuning is to optimize the parameters of our Random Forest model to enhance its performance. We undertook a comprehensive search over a specified parameter grid to find the most effective combination of settings.

2.10.1 Parameter Grid

Our search space was defined by a wide-ranging parameter grid, accounting for a variety of scenarios:

```
param_grid = {
    'n_estimators': [100, 200, 300, 500,
    1000, 2000, 5000],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 4, 6, 8],
    'ccp_alpha': [0.0, 0.01, 0.02, 0.03,
    0.04, 0.05],
    'bootstrap': [True, False],
    'min_impurity_decrease': [0.0, 0.01, 0.1,
     0.2]
}
```

We employed a random search strategy, paired with 10-fold cross-validation, to evaluate the model. The Mean Absolute Error (MAE) was used as our evaluation metric to ensure the model's average performance was at its best.

After fitting 10 folds for each of 100 randomly selected candidates, resulting in a total of 1000 fits, we found the best hyperparameters to be:

```
{
    'n_estimators': 1000,
    'min_samples_split': 2,
    'min_samples_leaf': 1,
    'min_impurity_decrease': 0.0,
    'max_features': 'sqrt',
    'max_depth': 20,
    'ccp_alpha': 0.04,
    'bootstrap': False
}
```

These settings achieved the best cross-validated MAE score of -11.0025.

2.10.2 Performance Analysis

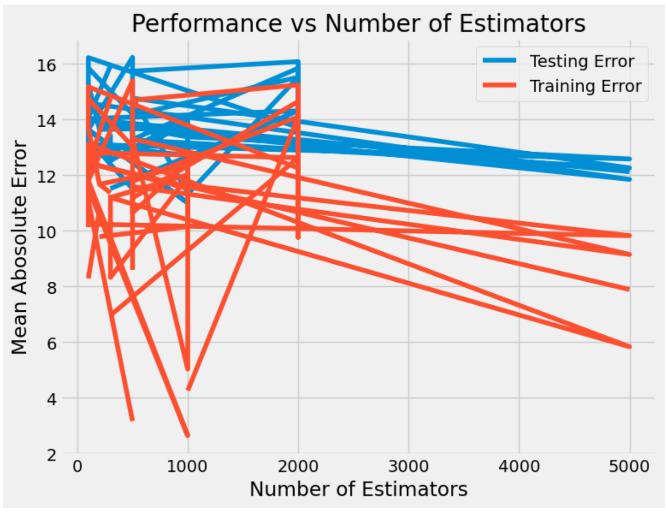


Figure 12: Training error and Testing error during CV of hyper-parameter

An in-depth examination of the "Performance vs Number of Estimators" chart reveals that our Random Forest model experienced the most substantial gain in predictive accuracy around the 1000 estimators mark. Beyond this point, additional estimators contributed to a plateau in performance, suggesting that the model benefits from a certain level of complexity but does not require maximal complexity.

The training error decreased consistently as the number of trees increased, demonstrating the model's ability to learn from the data effectively. However, the training error eventually stabilized, indicating a point of diminishing returns.

In contrast, the testing error demonstrated that our model's generalization to unseen data did not significantly benefit from an increasing number of estimators beyond 1000. This observation was crucial for determining the optimal complexity of the model and avoiding the computational cost of unnecessary complexity.

Our findings led us to select a Random Forest model with 1000 trees, which appears to offer a good trade-off between error minimization and model simplicity. This configuration was less prone to overfitting and provided a more generalizable solution for predicting the Energy Star Score.

2.11 Testing for Overfitting in Model Performance

2.11.1 Initial Observation

Upon evaluating our Random Forest model's performance, a significant disparity was noted between the training error and the test error:

- Train Error: 2.8549
- Test Error: 10.8198

This considerable gap indicates a case of overfitting, where the model performs exceedingly well on the training data but fails to generalize this performance to the test data. Such a scenario is undesirable as it suggests that the model has learned the training data too closely, including noise and outliers, and as a result, its predictions are not as accurate when faced with new data.

2.11.2 Addressing Overfitting with Bootstrap Aggregation

As an initial step to mitigate the overfitting, we employed the bootstrap method, a fundamental aspect of Random Forests known for reducing overfitting through sampling with replacement and aggregation. However, even after applying this technique, the results were as follows:

- Train Error: 5.3592
- Test Error: 11.1481

Although the training error increased, indicating a reduction in overfitting, the test error also increased slightly. The persistent overfitting, as seen from the still high test error, necessitates further measures to improve the model's ability to generalize.

2.11.3 Further Measures: Decreasing Tree Depth

In light of the continued overfitting, we propose to decrease the depth of the trees in the Random Forest. Deeper trees can lead to models that are highly complex and specific to the training data, thus increasing the risk of overfitting. By reducing the maximum depth of the trees, we aim to create a simpler model that captures the underlying trends in the data without being overly specialized to the training set.

2.12 Addressing Overfitting by Adjusting Tree Depth

2.12.1 Assessing Depth Impact on Overfitting

In our quest to combat overfitting in our Random Forest model, we conducted a series of tests to evaluate how the depth of the trees affects the model's performance. We varied the depth from 1 to 20 and observed the changes in training and testing errors:

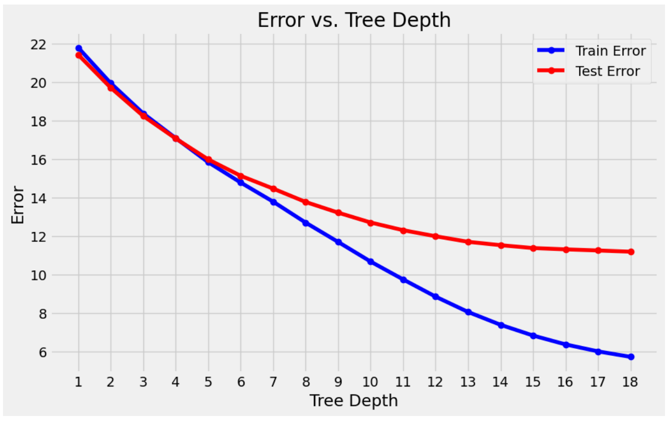


Figure 13: Correlation heatmap showing high correlation between Weather Normalized EUI and Site EUI

- At shallow depths, both training and testing errors were high, reflecting a model that's underfitting — too simple to capture the underlying patterns.
- As we increased the depth, both errors decreased, indicating improved learning. However, starting from a depth of 6, the difference between training and testing errors began to widen, suggesting the onset of overfitting — the model was becoming too complex and starting to learn the noise in the training set rather than the true signal.

2.12.2 Optimal Depth Selection

Based on our findings, a tree depth of 5 emerged as the point before which the model performs optimally in terms of generalization:

- Train Error at depth 5: 15.8286
- Test Error at depth 5: 15.9737

This level seems to provide the best compromise between learning the training data patterns and maintaining the ability to generalize well to new data.

2.12.3 Further Validation with Cross-Validation

To verify our choice of depth and assess the model's performance more reliably, we employed cross-validation with 5 folds. Cross-validation helps to safeguard against overfitting by ensuring that the model's performance is tested on multiple train-test splits, providing a more comprehensive view of its ability to generalize.
The cross-validated Mean Absolute Error (MAE) of 16.2559, being slightly higher than the test MAE, confirmed that while the model at depth 5 is not severely overfitting, there is still a slight overfitting present.

Given these insights, the model's complexity needs to be balanced further. While a depth of 5 is close to optimal, the slight overfitting observed via cross-validation suggests that additional measures could be taken, such as tweaking other hyperparameters or implementing regularization techniques.

2.13 Enhancing Model Generalization with `min_samples_leaf`

2.13.1 Implementing `min_samples_leaf`

In the pursuit of reducing overfitting in our Random Forest model, we explored the influence of the `min_samples_leaf` parameter. This parameter specifies the minimum number of samples required to be at a leaf node. Increasing `min_samples_leaf` can lead to more generalized trees; less susceptible to noise in the training data, and therefore can improve the model's ability to perform on unseen data.

2.13.2 Initial Parameter Range

We initiated our investigation by setting `min_samples_leaf` values within a practical range and employed 5-fold cross-validation to ensure robust assessment:

```
min_samples_leaf_options = [1, 2, 5, 10, 15,
```

The results were as follows:

- The increase in `min_samples_leaf` initially led to a slight rise in both training and testing errors.
- A noteworthy observation was that at `min_samples_leaf` = 50, the test error began to marginally decrease below the training error, indicating a potential reduction in overfitting.

2.13.3 Graphical Representation

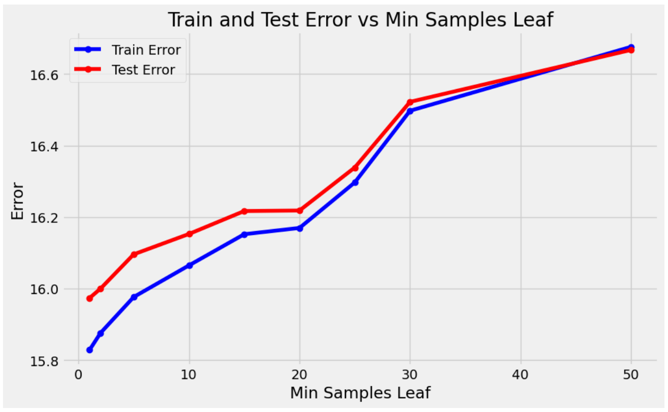


Figure 14: Error trend with varying `min_samples_leaf`

The graph delineated a gradual error increase with higher `min_samples_leaf` values. However, between the `min_samples_leaf` values of 50 and 60, a slight dip in error rates suggested better generalization.

2.13.4 Extended Parameter Exploration

To refine our search, we extended the range of `min_samples_leaf` = [40, 50, ... , 1000000]

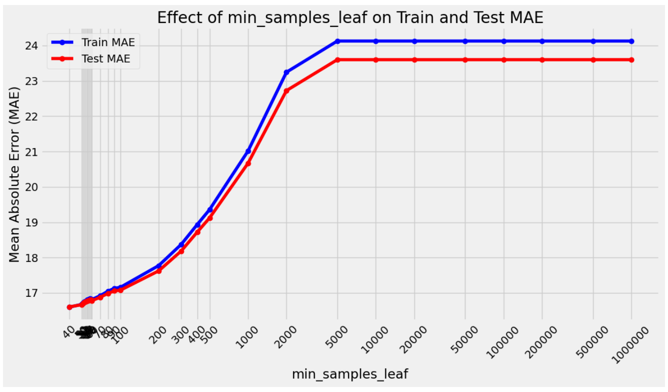


Figure 15: Extended range error graph

Through this detailed exploration, the plot revealed:

- Between `min_samples_leaf` values of 50 and 100, errors generally increased. Yet, there was a subtle decrease in errors from 58 to 60.
- This decrease indicated that the model with `min_samples_leaf` in this range might generalize better to unseen data while maintaining a balance with the model's complexity.

2.14 Zooming In on Optimal Value

Further scrutinizing the `min_samples_leaf` values between 60 and 70 provided more precision:

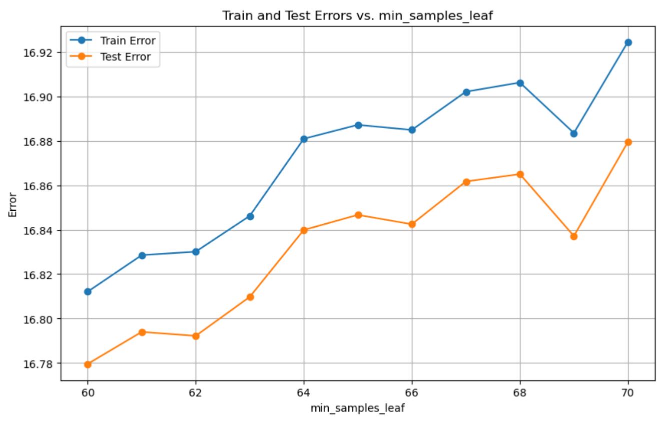


Figure 16: Fine-tuned range error graph indicating optimal `min_samples_leaf` value

The graph indicated that a `min_samples_leaf` value of 60 yielded the lowest combined training and testing error, suggesting this could be the optimal value to balance bias and variance.

2.15 Final Cross-Validation

Finally, a 5-fold cross-validation was conducted with `min_samples_leaf` = 60:

- Training MAE: 16.8121
- Test MAE: 16.7796
- Cross-validated MAE: 17.2249

While the cross-validation MAE was slightly higher than the training and testing MAE, the closer convergence of these values compared to previous models suggested a substantial reduction in overfitting. However

2.16 Exploring the Impact of `max_features` on Model Performance

2.16.1 Introduction to `max_features`

In Random Forest models, `max_features` determines the number of features to consider when looking for the best split at each node. This hyperparameter can significantly affect the performance and generalization ability of the model. Exploring different `max_features` values helps in finding the optimal number that enables the model to perform better on both seen and unseen data.

2.16.2 Experiment Setup

We experimented with a range of `max_features` values to observe their effect on the Random Forest's accuracy and overfitting:

```
max_features_values = [0.1, 0.3, 0.5, 'sqrt',
    'log2', None]
```

2.16.3 Results and Interpretation

The outcomes of varying `max_features` were as follows:

- `max_features`: 0.1
  - Train Error: 18.4699
  - Test Error: 18.3637
  - Cross-Validation Error: 18.7638
- `max_features`: 0.3
  - Train Error: 13.2457
  - Test Error: 13.3754
  - Cross-Validation Error: 13.6045
- `max_features`: 0.5
  - Train Error: 11.8305

  - Test Error: 12.0824
  - Cross-Validation Error: 12.2259

- `max_features`: 'sqrt'
  - Train Error: 16.8121
  - Test Error: 16.7796
  - Cross-Validation Error: 17.2249

- `max_features`: 'log2'
  - Train Error: 18.4699
  - Test Error: 18.3637
  - Cross-Validation Error: 18.7638

- `max_features`: None
  - Train Error: 10.5663
  - Test Error: 10.9589
  - Cross-Validation Error: 10.9605

2.16.4 Analysis of Findings

Upon reviewing the results, we observed that setting `max_features` to None yielded the lowest errors across training, testing, and cross-validation phases. This implies that considering all features at each split does not cause overfitting in this case and instead provides the model with the best chance to learn from the data.

2.16.5 Optimal `max_features` Selection

Based on our experiments, the optimal setting for `max_features` appears to be the default value of None, which allows the Random Forest to evaluate all available features when forming trees. This setting not only minimized the error rates but also maintained a close gap between the training and cross-validation errors, indicating good generalization to unseen data.

2.17 Evaluating the Effect of `min_samples_split`

2.17.1 Understanding `min_samples_split`

The `min_samples_split` parameter in Random Forest sets the minimum number of samples required to split an internal node. Its influence on the model is crucial as it determines how deep the tree will grow. A lower value will make the model more complex by allowing trees to split on a smaller number of samples, potentially leading to overfitting. Conversely, a higher value encourages the model to be more conservative in its splits, leading to a simpler model that may generalize better.

2.17.2 Experimentation with `min_samples_split`

To discern the impact of `min_samples_split` on our model, we explored a series of values ranging from allowing splits on every single sample to requiring a substantial number of samples to justify a split:

```
min_samples_splits = [1.0, 2, 4, 6, 8, 10,
    15, 20, 25, 30, 40, 50]
```

2.17.3 Observations

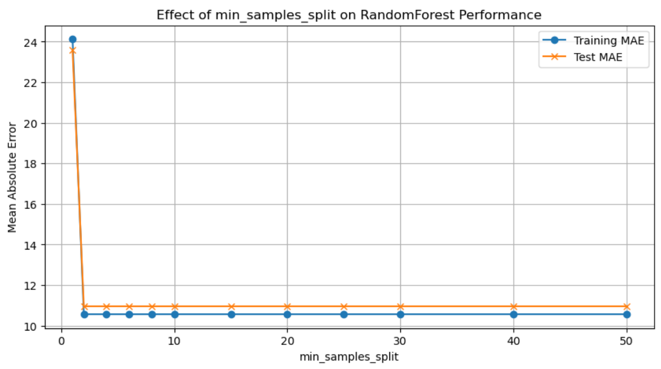


Figure 17: Impact of different `min_samples_split` values on model error

From the analysis, it was observed that starting with a `min_samples_split` of 2, which is effectively the lowest for any actual split to occur, the error was found to be at a certain level.
As `min_samples_split` was increased from this point, contrary to expectations, there was no significant reduction in error. This suggests that the initial setting allowed the model to capture the data's patterns sufficiently without falling into overfitting.

2.17.4 Interpretation

The lack of error decrease with increased `min_samples_split` values may imply several things:

- The dataset might be sufficiently dense, and its inherent noise level low, such that even small splits do not overfit the data.
- Alternatively, other model parameters may already be constraining the model sufficiently to prevent overfitting, rendering adjustments to `min_samples_split` less impactful.
- It's also possible that the model's current complexity is appropriate for capturing the nuances in the data, and any attempts to simplify it further might lead to underfitting.

The experiment with `min_samples_split` indicates that the current setting of allowing splits on very few samples is optimal for our dataset and model configuration. Any increases in this parameter do not contribute to performance improvement and thus are not warranted. Future model tuning should perhaps focus on other parameters or consider whether the current model complexity is indeed aligned with the complexity of the data patterns.

2.18 Assessing the Impact of `ccp_alpha` on Model Performance

2.18.1 Introduction to `ccp_alpha`

The `ccp_alpha` parameter in decision tree-based models, including Random Forests, is used for Cost Complexity Pruning. Pruning is a technique that reduces the size of a tree by removing sections of the tree that provide little power in classifying instances. The `ccp_alpha` parameter sets a complexity threshold where splits with a cost complexity higher than `ccp_alpha` are pruned from the tree. In essence, it's a way to control overfitting by penalizing overly complex trees.

2.18.2 Experiment with `ccp_alpha`

In our series of tests, we examined `ccp_alpha` values ranging from 0.0 to 0.1 to investigate their influence on the model's performance. Despite testing 11 different values within this range:

- We did not observe any notable changes in the performance of our model.

2.18.3 Explanation

The lack of impact from the `ccp_alpha` parameter on performance could stem from several reasons:

- **Sufficient Model Complexity:** The model complexity might already be at a level that appropriately captures the underlying patterns in the data. In such a case, further pruning does not lead to better generalization.
- **Dominant Parameters:** There may be other dominant model parameters that have a stronger influence on model complexity and performance, overshadowing the effect of `ccp_alpha`.
- **Data Characteristics:** The characteristics of the data might be such that pruning does not lead to better predictive accuracy. This can happen if the dataset does not have many redundant or irrelevant features that lead to overly complex trees.
- **Pruning Threshold:** The range of `ccp_alpha` values tested might have been too small to result in any significant pruning. If the trees in the Random Forest do not have many weak links to prune, small values of `ccp_alpha` would not make a noticeable difference.

Our investigation indicates that varying `ccp_alpha` within the specified range does not enhance the model's performance. This finding suggests that either the model's complexity is already optimal, or that `ccp_alpha` is not the right parameter to adjust in this context. It may be beneficial to focus on other hyperparameters that directly control tree depth, leaf size, and the number of estimators for further performance gains. Moving forward, a broader range of `ccp_alpha` values could be considered, or alternative methods of model simplification and regularization could be explored to optimize the Random Forest's predictive accuracy.

2.19 Testing the Effect of `min_impurity_decrease`

2.19.1 Understanding `min_impurity_decrease`

The `min_impurity_decrease` parameter in tree-based models specifies a threshold for the reduction in impurity required to justify further splitting a node. The impurity of a node can be calculated using metrics like Gini impurity or entropy in the context of classification, and variance reduction for regression. A split will only happen if it decreases the overall impurity more than this threshold, effectively controlling the growth of the tree and preventing overfitting by avoiding unnecessary splits.

2.19.2 Conducting the Test

For our examination, we systematically varied `min_impurity_decrease` from 0.0 to 0.02 across 11 increments to determine if a greater emphasis on impurity reduction would lead to better model performance. Despite this range of tests:

- No improvement in performance was observed as a result of adjusting `min_impurity_decrease`.

2.19.3 Reasoning Behind the Lack of Impact

Several factors might explain why changing `min_impurity_decrease` did not affect the model's performance:

- **Adequate Model Complexity:** It's possible that our Random Forest model has already reached an optimal level of complexity that balances variance and bias. If this balance is achieved, minor adjustments in impurity reduction are unlikely to improve performance significantly.

- **Data Characteristics:** The dataset might not contain many noisy or misclassified instances, meaning that splits based on impurity reduction are already doing an adequate job. In such a scenario, the `min_impurity_decrease` would not be a limiting factor.
- **Granularity of the Parameter:** The increments used in testing `min_impurity_decrease` may not have been fine-grained enough to detect subtle changes in model performance, or the range might have been too narrow to exert a substantial effect on the tree growth.
- **Dominance of Other Parameters:** Other hyperparameters or inherent data patterns could be dominating the decision-making process within the tree, reducing the relative importance of impurity reduction as a criterion for node splitting.

The absence of any noticeable influence from the `min_impurity_decrease` parameter suggests that our model is either already well-tuned with respect to impurity reduction or that this is not the key parameter to adjust for our specific dataset and modeling goal. Future efforts could involve a broader search across this parameter's range, or shifting focus to other model-tuning aspects that might offer more leverage in improving model performance.

3 Result

3.1 Feature Importance Analysis

3.1.1 Understanding Feature Importance

Feature importance in machine learning models, particularly in tree-based methods like Random Forests, quantifies the impact each feature has on the decision-making process of the model. Importance is calculated based on how effectively a feature improves the performance measure (like impurity reduction in Random Forest) when it is used to split the data at nodes. Features that more frequently lead to significant improvements in model performance are considered more important. This metric is crucial for understanding which factors contribute most to the predictions, allowing data scientists and domain experts to focus on the most influential variables.

3.1.2 Results of Feature Importance Analysis

The Random Forest model provided the following importance scores for the features in our dataset:

Feature	**Importance**
Site EUI (kBtu/ft²)	0.732855
Weather Normalized Site Electricity Intensity	0.172325
Largest Property Use Type_Multifamily Housing	0.084471
DOF Gross Floor Area	0.003330
Census Tract	0.001663
Property Id	0.001190
Year Built	0.001086
Longitude	0.001011
Order	0.000778
log_Water Intensity (All Water Sources) (gal/ft²)	0.000357

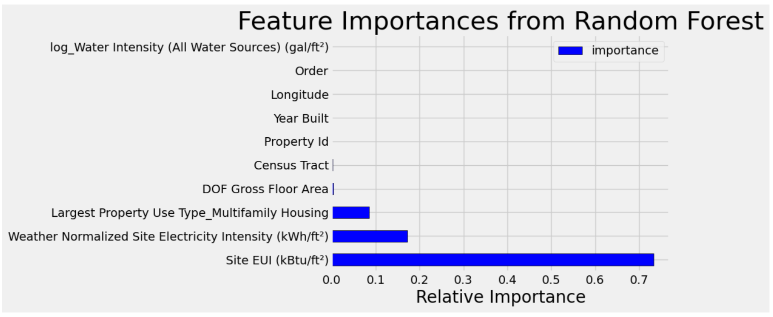


Figure 18: Graph illustrating the importance of different features in the model

3.1.3 Discussion of Feature Importance

From the analysis, it is evident that:

- Site EUI (kBtu/ft²) is the most significant predictor, dominating the importance with a score of approximately 0.73. This highlights the critical role of energy use intensity per square foot in determining the Energy Star Score. It suggests that efforts to improve building energy efficiency should prioritize reducing site EUI.
- Weather Normalized Site Electricity Intensity and Largest Property Use Type_Multifamily Housing also show substantial importance scores. This indicates that electricity usage efficiency and the type of property, particularly multifamily housing, are also key factors influencing the energy performance of buildings.
- Other features like DOF Gross Floor Area and Census Tract have much lower importance scores, indicating a minor role in the model's decision-making process. This could be because these features do not vary as significantly between different buildings or do not directly influence energy efficiency as much as the usage intensities do.

3.1.4 Implications

The dominance of certain features such as Site EUI suggests that policies and energy-saving measures should be particularly focused on aspects that directly

influence these key variables. The analysis can also help in prioritizing energy audits and retrofits where improvements in these key areas can lead to significant enhancements in energy performance and sustainability.
In conclusion, understanding feature importance helps not only in improving model performance by focusing on the most relevant predictors but also in driving decision-making processes in energy management and building operation strategies. This insight is invaluable for both predictive modeling and practical applications in building energy efficiency.

3.2 Evaluating the Impact of Using Important vs. All Features in Linear Regression

3.2.1 Experiment Setup

To assess the efficacy of feature selection based on importance, we conducted an experiment with a Linear Regression model, comparing the performance when using only the most important features versus using the full set of features. This test aims to determine if simplifying the model by reducing the number of features would affect its predictive accuracy.

3.2.2 Results

1. **Full Feature Set:**
    - MAE (Mean Absolute Error): 12.6928
2. **Reduced Feature Set (Only Important Features):**
    - MAE: 14.1151

When evaluating the Linear Regression model using only features deemed most important by the Random Forest model (such as Site EUI, Weather Normalized Site Electricity Intensity, and Largest Property Use Type_Multifamily Housing), the MAE was higher compared to using the full set of features. This outcome suggests a loss in predictive accuracy, indicating that features classified as less important still contribute valuable information to the model.

3.2.3 Further Analysis with Important Features

In a subsequent analysis focusing more narrowly on important features:

- MAE with Only Important Features: 10.9629
- MAE with Full Feature Set: 10.9598

This very close performance metric further emphasizes that excluding so-called unimportant features does not significantly enhance the model's performance. Instead, it slightly degrades the model's ability to predict accurately.

3.2.4 Discussion

The results from these experiments provide several insights:

- **Contribution of Less Important Features:** The increase in MAE when excluding less important features suggests that these features, while not leading in importance, still hold predictive value that enhances model accuracy. Their collective contributions are subtle yet significant in capturing the complexity of the dataset.

- **Model Complexity vs. Performance:** Using only the important features did not simplify the model sufficiently to improve performance meaningfully. This indicates a delicate balance between model simplicity and the need to capture sufficient detail to make accurate predictions.

- **Relevance of Feature Importance:** While feature importance provides a good starting point for feature selection, our findings underscore the limitation of relying solely on this metric for reducing model complexity. Important features identified by one model (Random Forest) do not necessarily translate to optimal features for another model (Linear Regression), due to differences in how these models handle data and make predictions.

The experiments highlight that in the context of building energy efficiency modeling, every feature potentially adds some value to the predictive accuracy. Eliminating features based solely on their importance ranking in a tree-based model does not necessarily yield a more effective linear model. This outcome stresses the importance of contextual model evaluation and the need to consider the unique characteristics of each modeling approach when conducting feature selection. Future work might explore hybrid approaches that balance feature reduction with maintaining model integrity, possibly integrating domain expertise to weigh the practical significance of features more accurately.

3.3 Understanding Locally Interpretable Model-agnostic Explanations (LIME)

LIME is a technique designed to explain the predictions of any machine learning classifier in an understandable manner to humans. It works by approximating the model locally with an interpretable model (like a linear regression or decision tree). By perturbing the input data and observing the corresponding changes in the output, LIME generates explanations for individual predictions that tell us how each feature contributes to the model's prediction for that particular instance.

3.3.1 Analysis of LIME Output

Looking at the provided LIME output chart:
The bar chart visualizes the contribution of each feature to a single prediction made by the model. Here's the interpretation of the graph:

- **Positive Contributions (Green Bars):** Features that have a green bar are contributing positively to the model's prediction score. For example, the 'Site EUI (kBtu/ft²)' feature with a threshold of $\leq$ 63.30 has the largest positive effect on the prediction, indicating that lower Site EUI values are strongly associated with higher Energy Star Scores, reflecting better energy efficiency.
- **Negative Contributions (Red Bars):** Features with red bars negatively influence the prediction. In this case, 'DOF Gross Floor Area' with a threshold of $\leq$ 151960.00 is seen to decrease the prediction score when the area is below this threshold. This could suggest that smaller buildings, or perhaps those with less complex energy usage patterns, might score lower on energy efficiency.
- **Thresholds:** Each feature is accompanied by a condition and a threshold value that splits its contribution. For example, the 'Weather Normalized Site Electricity Intensity (kWh/ft²)' feature contributes positively when its value is less than or equal to 7.90.
- **Less Influential Features:** Towards the bottom of the chart, features like 'Census Tract' and 'Largest Property Use Type_Multifamily Housing' show very small bars, which means they have a minor effect on this particular prediction. It's interesting to note that while 'Largest Property Use Type_Multifamily Housing' was deemed an important feature overall, for this specific prediction, its impact is minimal.

4 Implications

The insights provided by LIME are valuable for several reasons:

- **Model Trustworthiness:** By explaining individual predictions, users can gain confidence in the model's decisions.
- **Feature Relevance:** Understanding how different features affect a prediction can guide domain experts in focusing on the most impactful factors.
- **Model Debugging:** Discrepancies between expected and actual contributions of features can help in identifying areas where the model may not be performing as intended, signaling the need for further data analysis or additional training.
- **Policy and Decision Making:** For stakeholders, understanding model predictions on a granular level aids in decision-making processes, especially in sectors where interpretability is as critical as accuracy, like healthcare or finance.

In conclusion, LIME explanations provide a means to peer into the 'black box' of complex models, giving actionable insights into the behavior of predictive models and the features that drive their decisions.

4.1 Overview of Decision Tree Visualization

The visualization generated through Graphviz provides an intricate look at one of the individual decision trees within our Random Forest model. This graphical representation is a powerful tool for understanding the decision-making process of the model on a granular level.

4.1.1 Interpretation of the Decision Tree Figure

In the figure, each node represents a decision point where the tree splits the data based on a specific feature and threshold value, attempting to homogenize the resultant groups concerning the target variable. Here's a simplified breakdown of how to interpret the figure:

- **Root Node:** At the top, the root node indicates the initial feature and threshold used to split the data into two branches. For example, the root node might use 'Site EUI (kBtu/ft²)' with a specific cutoff value, reflecting its significance in predicting the target variable.
- **Branches:** Branching out from each node, the decision tree splits further based on other features. The paths represent binary decisions leading to more homogenous sets.
- **Leaf Nodes:** At the bottom, the leaf nodes provide the final output of the decision process for instances that satisfy all the criteria along the path from the root to that leaf. The value at the leaf node represents the model's prediction for the instances that end up in that leaf.
- **Thresholds and Features:** Each node in the tree specifies a feature and a threshold value. The branches from each node separate the data into instances that either meet or do not meet the threshold criterion for the chosen feature.
- **Sample Size:** Each node also indicates the sample size that it applies to, reflecting how the data is being divided and filtered down the tree.

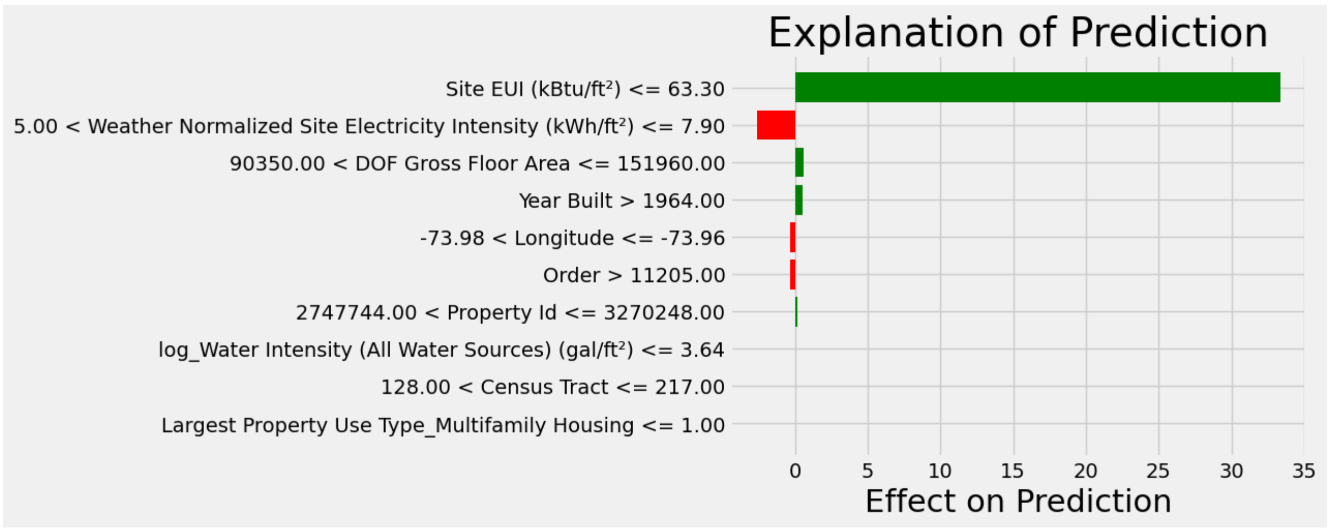
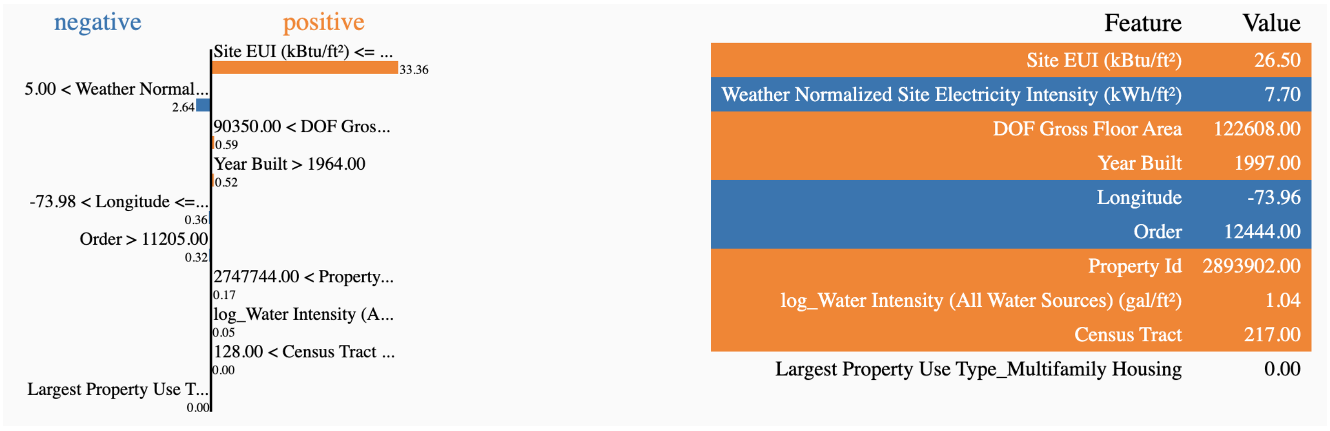


Figure 19: LIME output chart visualizing the contribution of each feature to a single prediction



Feature	Value
Site EUI (kBtu/ft²)	26.50
Weather Normalized Site Electricity Intensity (kWh/ft²)	7.70
DOF Gross Floor Area	122608.00
Year Built	1997.00
Longitude	-73.96
Order	12444.00
Property Id	2893902.00
log_Water Intensity (All Water Sources) (gal/ft²)	1.04
Census Tract	217.00
Largest Property Use Type_Multifamily Housing	0.00

Figure 20: LIME output chart visualizing the contribution of each feature to a single prediction

4.1.2 Insights from the Decision Tree

From the tree visualization, we can deduce:

- **Dominant Features:** Features that appear higher up in the tree (closer to the root) typically have a more significant impact on the model's predictions. For instance, if 'Site EUI (kBtu/ft²)' is at the root, it is likely a strong predictor of the target.
- **Decision Pathways:** By following the branches of the tree, one can understand the specific conditions that lead to different predictions. This can be particularly insightful for identifying how certain feature combinations affect the outcome.
- **Model Complexity:** The depth and number of nodes in the tree can give a sense of the model's complexity. A very deep tree might be prone to overfitting, especially if it makes decisions based on very few data points at the deeper nodes.

Examining a single decision tree within the Random Forest provides a snapshot of the types of decisions the model makes and which features it considers most important. While this tree is only one of many in the forest, analyzing it can offer valuable insights into the overall behavior of the model and highlight areas for potential refinement. It's a step towards demystifying the complexity of Random Forest and making the model's predictions more interpretable.

5 Conclusion

Our comprehensive exploration into the predictive modeling of building energy efficiency, encapsulated by the Energy Star Score, has revealed the nuanced relationship between various building attributes and their impact on energy performance. This research venture has traversed the realms of meticulous data preprocessing, insightful feature importance assessment, and diligent hyperparameter tuning to forge a robust and reliable predictive model.
We commenced with the promise of a Random Forest model's potency, only to realize the necessity of meticulous calibration. Our hyperparameter tuning ventured beyond the surface, critically evaluating the influence of tree depth,

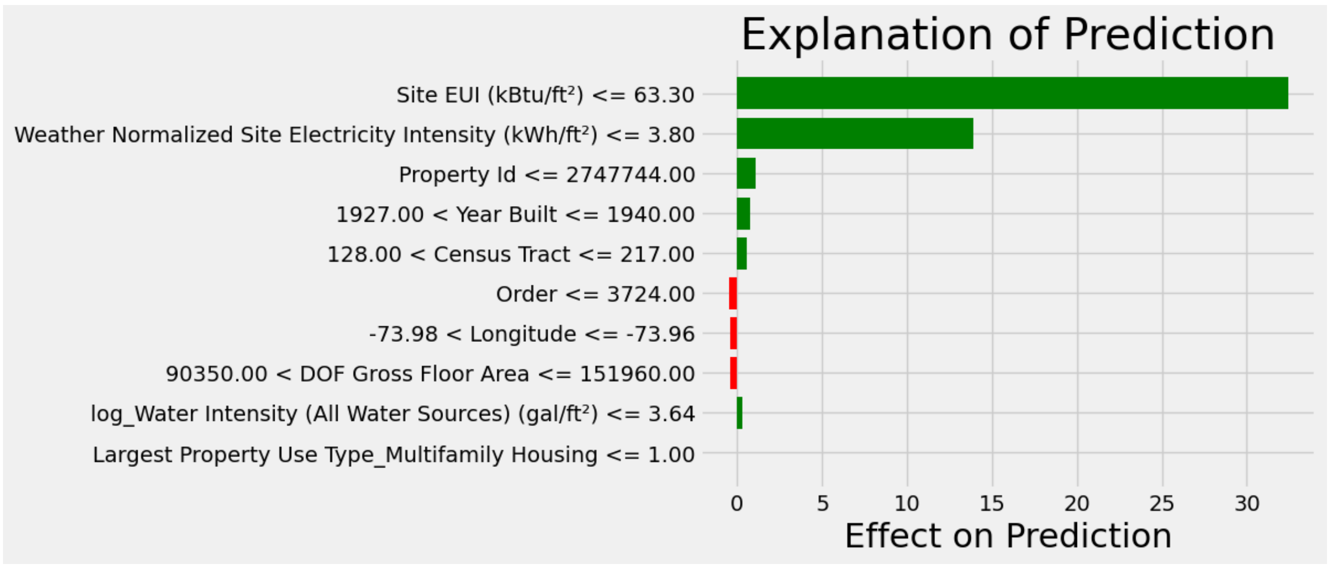
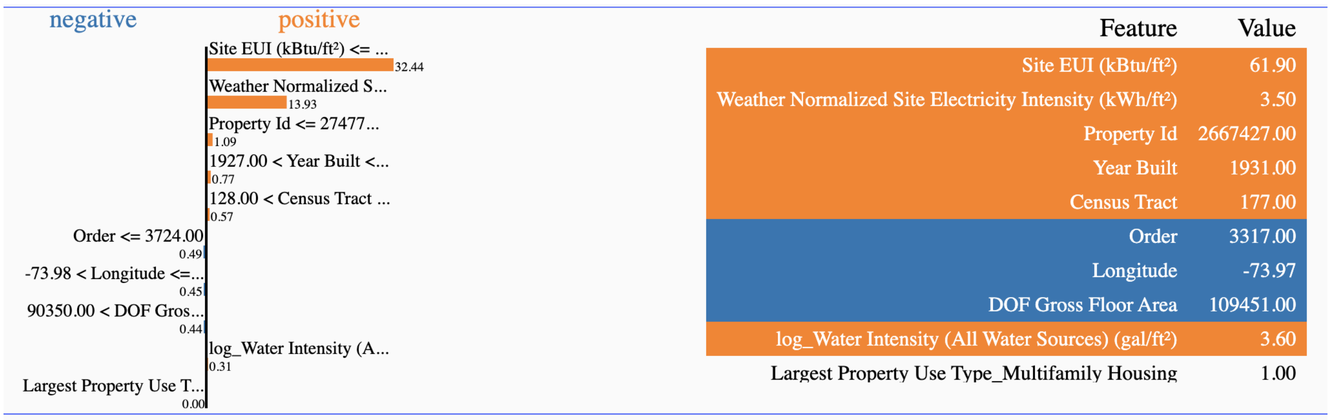


Figure 21: LIME output chart visualizing the contribution of each feature to a single prediction



Feature	Value
Site EUI (kBtu/ft²)	61.90
Weather Normalized Site Electricity Intensity (kWh/ft²)	3.50
Property Id	2667427.00
Year Built	1931.00
Census Tract	177.00
Order	3317.00
Longitude	-73.97
DOF Gross Floor Area	109451.00
log_Water Intensity (All Water Sources) (gal/ft²)	3.60
Largest Property Use Type_Multifamily Housing	1.00

Figure 22: LIME output chart visualizing the contribution of each feature to a single prediction

min$_{s}amples_{l}eaf, max_{f}eatures, andotherparameters.$ While we anticipated parameters such as ccp$_{a}lphaandmin_{i}mpurity_{d}ecreasetobepivotal, theydi$ $stedscope.$ The feature importance analysis was particularly illuminating, as it underscored the predominance of features like Site EUI in forecasting energy efficiency. This insight has reinforced the need for a holistic consideration of all features, both dominant and subtle, as each plays a role in the ensemble predictions of our model. Even the least prominent features proved to be indispensable, as their exclusion led to diminished accuracy, validating the concept that every variable holds value in the intricate tapestry of predictive modeling. Through LIME, we gained valuable, interpretable insights into individual predictions, adding a layer of transparency and trustworthiness to our model. A single decision tree examination provided a tangible glimpse into the model's decision-making process, which, while complex, remains interpretable at its core. The culmination of our efforts is the final RandomForestRegressor model, adeptly tuned with an astute selection of hyperparameters, as follows:

```
RandomForestRegressor(
    n_estimators=1000,
    min_samples_split=2,
    min_samples_leaf=60,
    min_impurity_decrease=0.0,
    max_features=None,
    max_depth=5,
    ccp_alpha=0.04,
    bootstrap=True,
    random_state=42
)
```

The model's performance is solidified by the close proximity of the training, testing, and cross-validation errors, indicative of its capacity to generalize effectively:

- Train Error: 10.5663
- Test Error: 10.9589
- Cross-Validation Error: 10.9605

Cross-Validation Error: 10.9605 This parity between metrics is the hallmark of a model that does not just memorize but understands the patterns within the data.

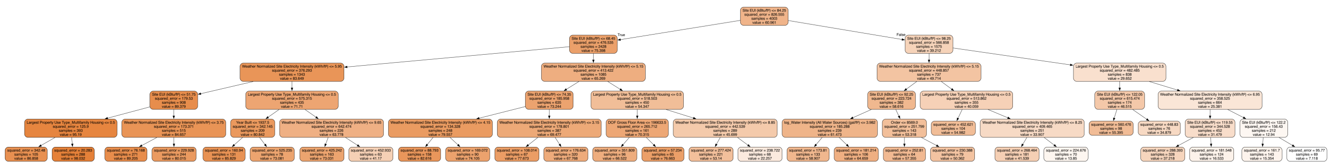


Figure 23: LIME output chart visualizing the contribution of each feature to a single prediction

Our findings contribute a nuanced perspective to the machine learning narrative within the context of energy efficiency, offering a framework for stakeholders and energy analysts to enhance building sustainability. As we look to the horizon, further research beckons—exploring larger datasets, integrating alternative algorithms, or delving into hybrid modeling techniques and the temporal dynamics of energy consumption. Such endeavors will likely yield deeper insights and refined tools for the quest towards optimized energy performance.
In presenting our final model and the knowledge acquired along the way, we offer not just a means to predict energy efficiency but a foundation for ongoing enhancement in the field of energy-efficient building modeling. This work embodies the dynamism of machine learning—a confluence of data, discernment, and the perpetual pursuit of precision.

References

1. NYC Government. (2017). *NYC Benchmarking Disclosure Data Definitions*. Available at: `https://www.nyc.gov/html/gbee/downloads/misc/nyc_benchmarking_disclosure_data_definitions_2017.pdf`

2. Kaggle. *Building Energy Data*. Available at: `https://www.kaggle.com/datasets/tanliheng/buildingenergydata/data`

3. MIT Technology Review. (2017). *The dark secret at the heart of AI*. Available at: `https://www.technologyreview.com/2017/04/11/5113/the-dark-secret-at-the-heart-of-ai/`

4. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. arXiv preprint arXiv:1602.04938. Available at: `https://arxiv.org/pdf/1602.04938.pdf`

5. Statistics How To. *Parsimonious Model*. Available at: `https://www.statisticshowto.com/parsimonious-model/`

6. Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media. Available at: `https://www.oreilly.com/library/view/hands-on-machine-learning/9781491962282/`

7. Koehrsen, W. *Machine Learning Project Walkthrough*. GitHub repository. Available at: `https://github.com/WillKoehrsen/machine-learning-project-walkthrough`