**Project Document: Household Energy Consumption Prediction**

**Objective:** The objective of this project is to develop a machine learning model that accurately predicts household energy consumption based on historical data. By leveraging this model, consumers will gain valuable insights into their energy usage patterns, allowing for more informed decisions about optimizing consumption. Additionally, energy providers can use the predictive model to forecast demand more effectively, enabling better resource planning and cost reduction. The ultimate goal is to deliver actionable insights into energy usage trends and provide a robust predictive model that can serve as a foundation for future research and development in energy management systems.

**Approach:**

**1.Data Preprocessing:**

**1. 1. Data Import and Preprocessing:**

The first step involved loading the dataset into a Jupyter Notebook to explore its structure, variables, and quality. The dataset contained 9 columns with the following data types:

**# Column Dtype**

0 Date object

1 Time object

2 Global\_active\_power object

3 Global\_reactive\_power object

4 Voltage object

5 Global\_intensity object

6 Sub\_metering\_1 object

7 Sub\_metering\_2 object

8 Sub\_metering\_3 float64

It was observed that several columns, including Date, Time,*Global\_active\_power*, *Global\_reactive\_power*, *Voltage*, *Global\_intensity*, *Sub\_metering\_1*, and *Sub\_metering\_2*, were incorrectly classified as object data types instead of numerical types.

To resolve this, the **Date** and **Time** columns were combined into a single **Datetime** column, allowing for more effective time-series analysis. Additionally, all other columns were converted to float to ensure accurate numerical computations.

**1.2 Handling Missing Values:**

The next step was to check for any missing values. The following results were found:

Datetime 0

Global\_active\_power 25979

Global\_reactive\_power 25979

Voltage 25979

Global\_intensity 25979

Sub\_metering\_1 25979

Sub\_metering\_2 25979

Sub\_metering\_3 25979

The dataset contained missing values in the measurements columns, with approximately 1.25% of the rows affected.

To handle these missing values, we used **median imputation**, as the median is less affected by outliers and provides a more robust measure for replacing missing data.

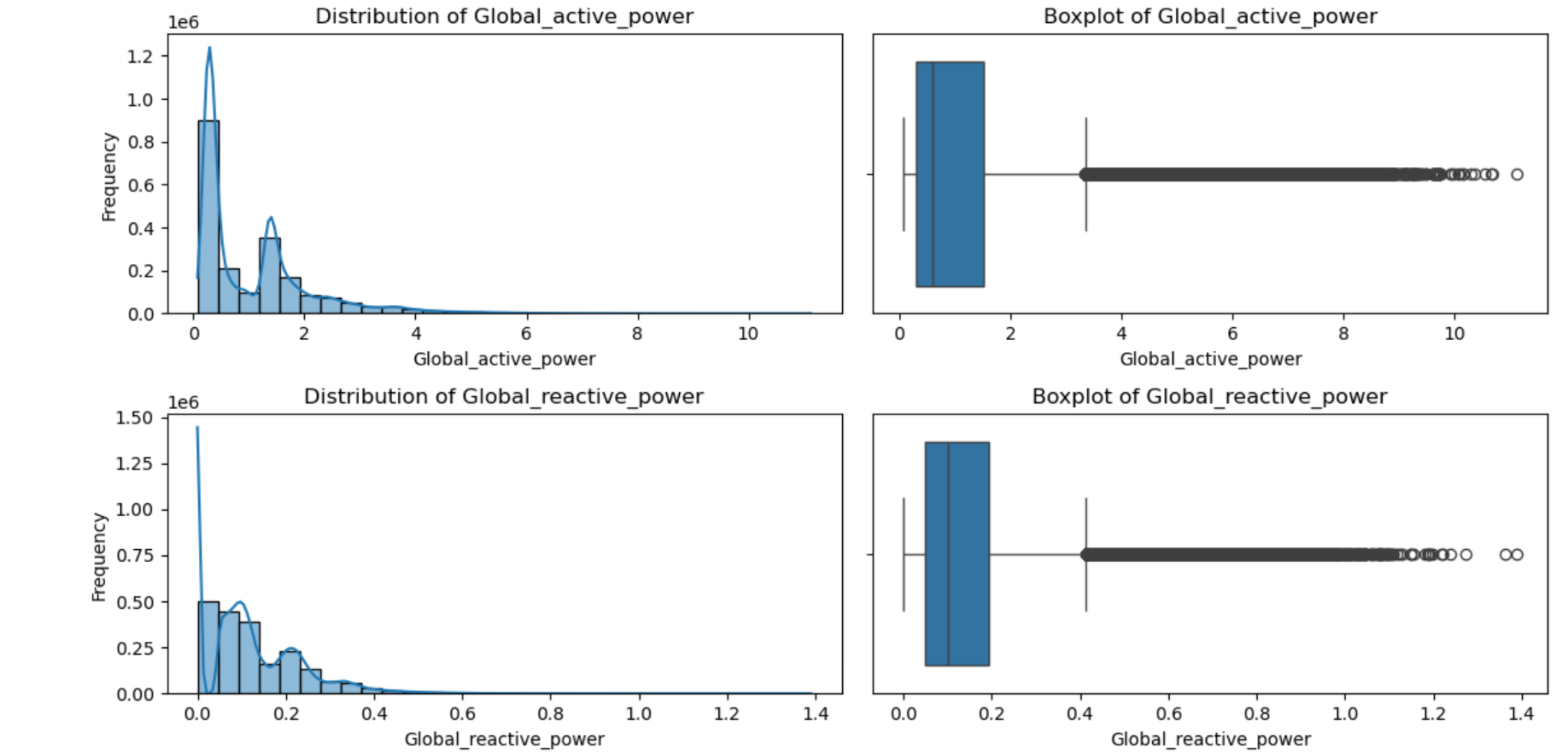
**1.3 Outlier Detection**

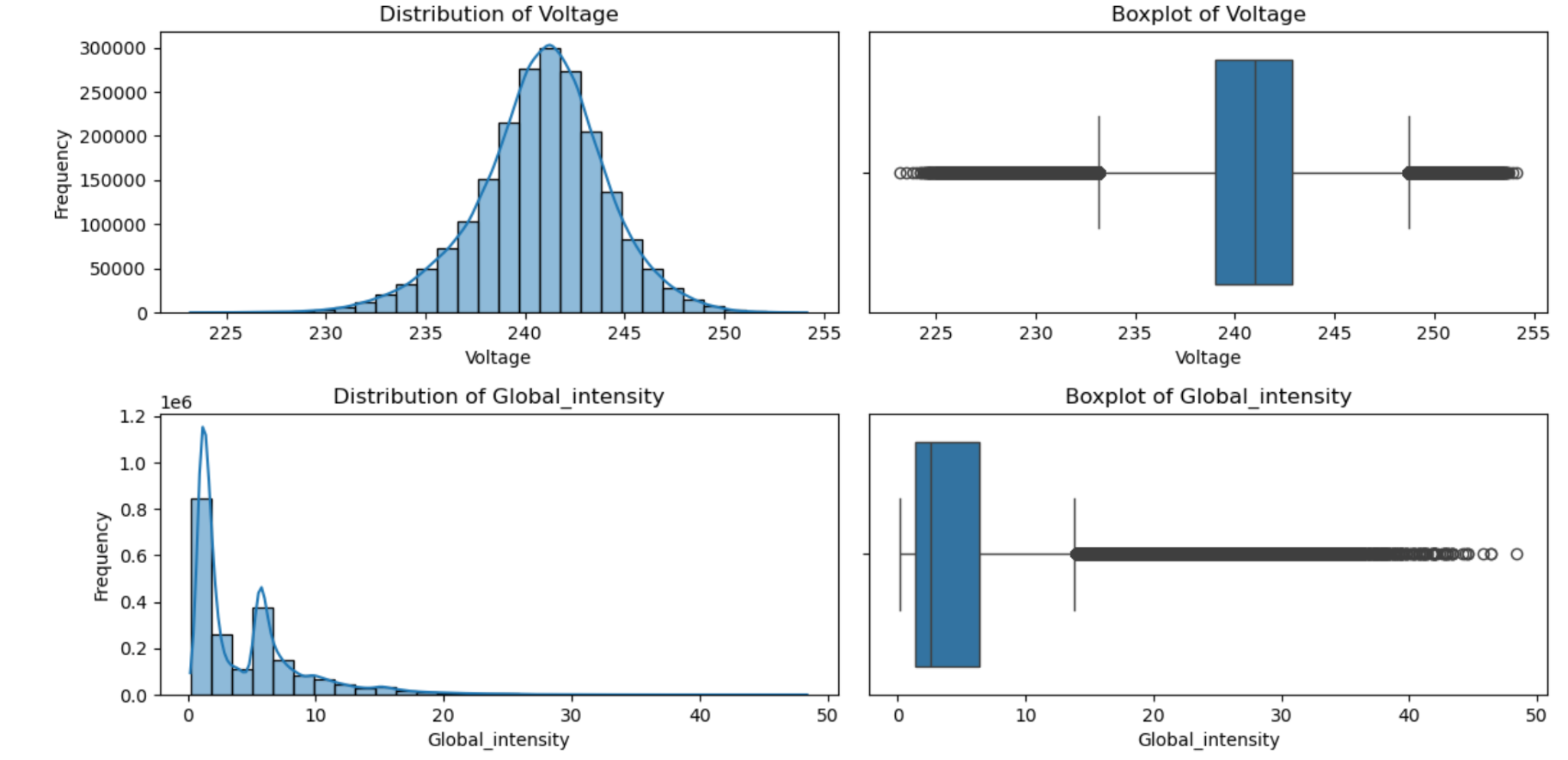
After handling missing values, outliers in the dataset were detected by visualizing the data using **boxplots** and **histograms**. The following observations were made:

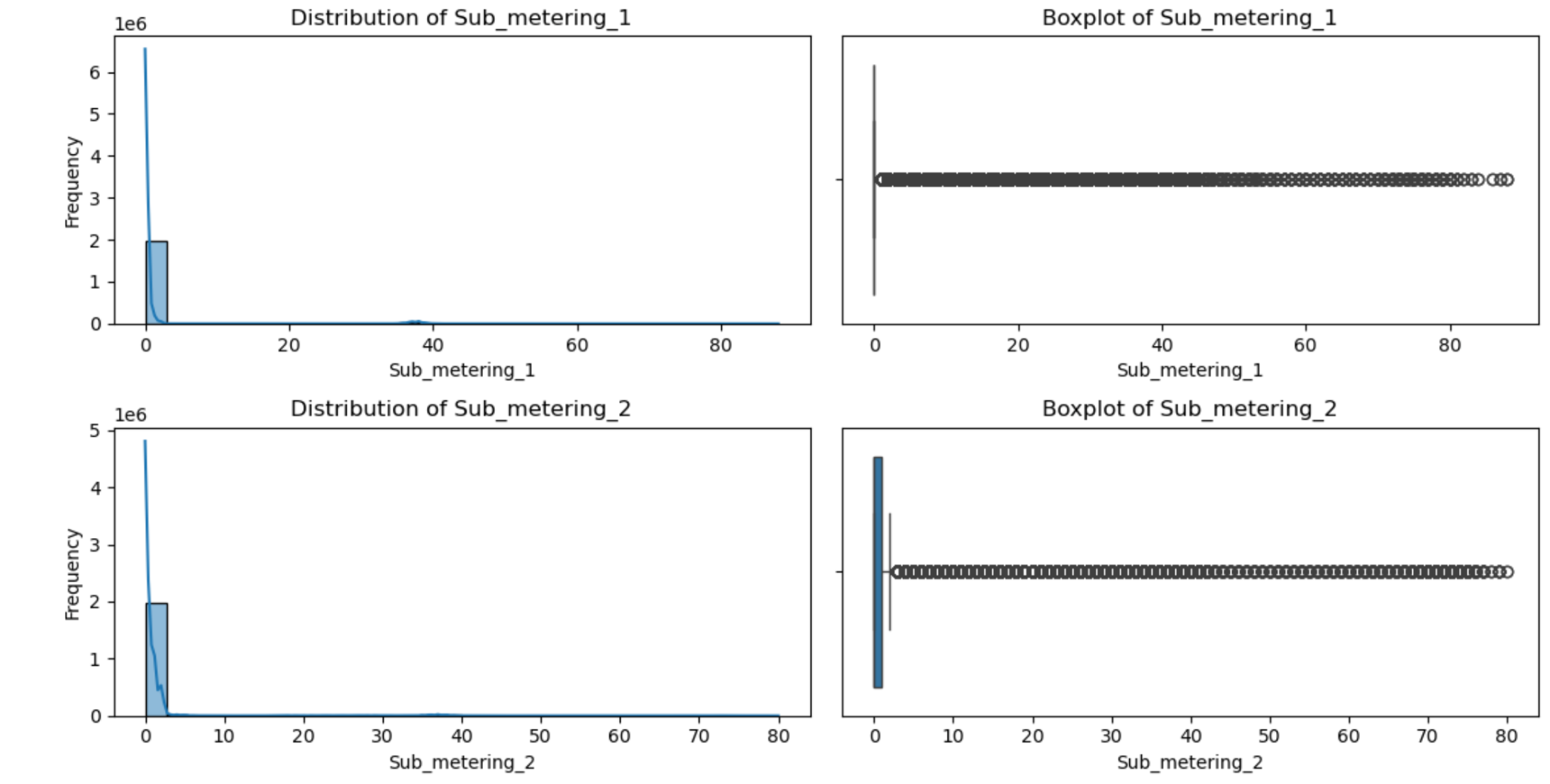
* **Global\_active\_power**, **Global\_reactive\_power**, **Global\_intensity**, **Sub\_metering\_1**, and **Sub\_metering\_2** exhibited right-skewed distributions, indicating the presence of outliers on the higher end of the data.
* The distribution of **Voltage** was approximately normal but had outliers on both ends, making it both right and left skewed.
* **Sub\_metering\_3** showed no significant outliers and was well-behaved compared to other variables.

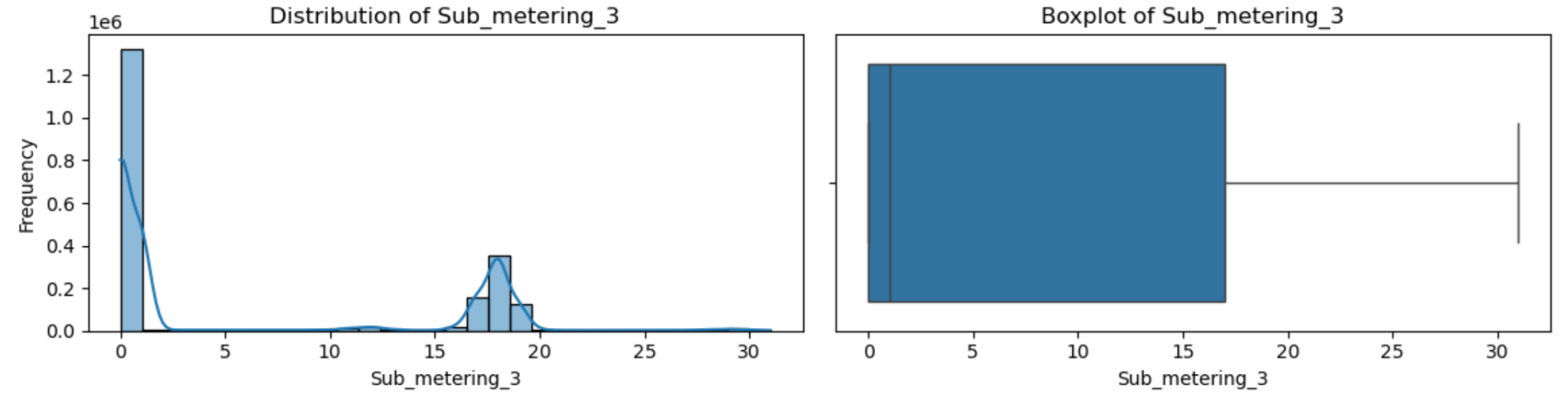
**Graphical Representation:**

Below is the graphical representation to check for outliers in the features

****

****

****

****

**1.4** **Outlier Treatment**

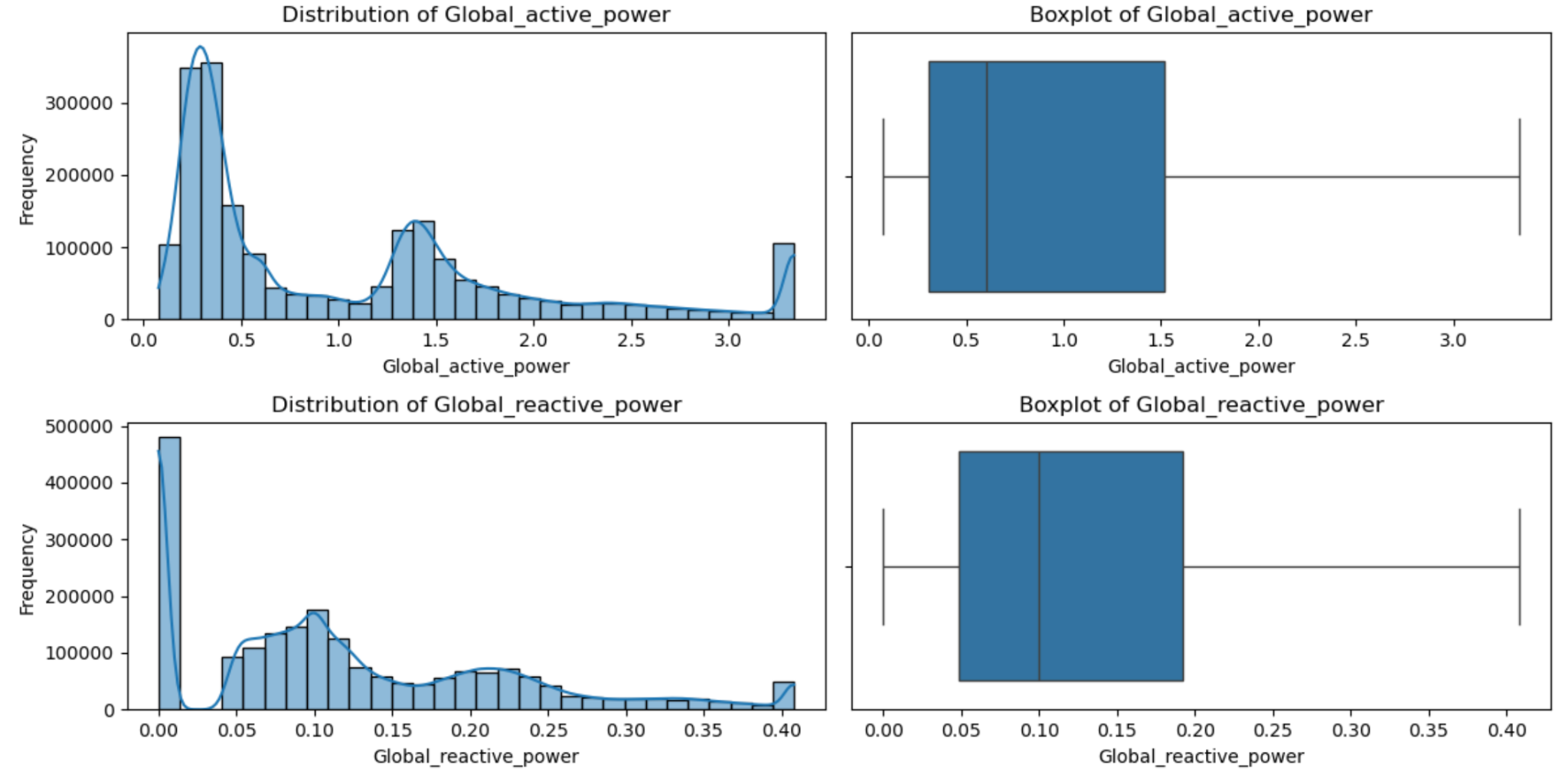
Outliers identified in the dataset were handled using two different techniques: the **Interquartile Range (IQR) method** and **Winsorization**, with the choice of method based on the type of skewness observed and the effectiveness of the technique for each feature.

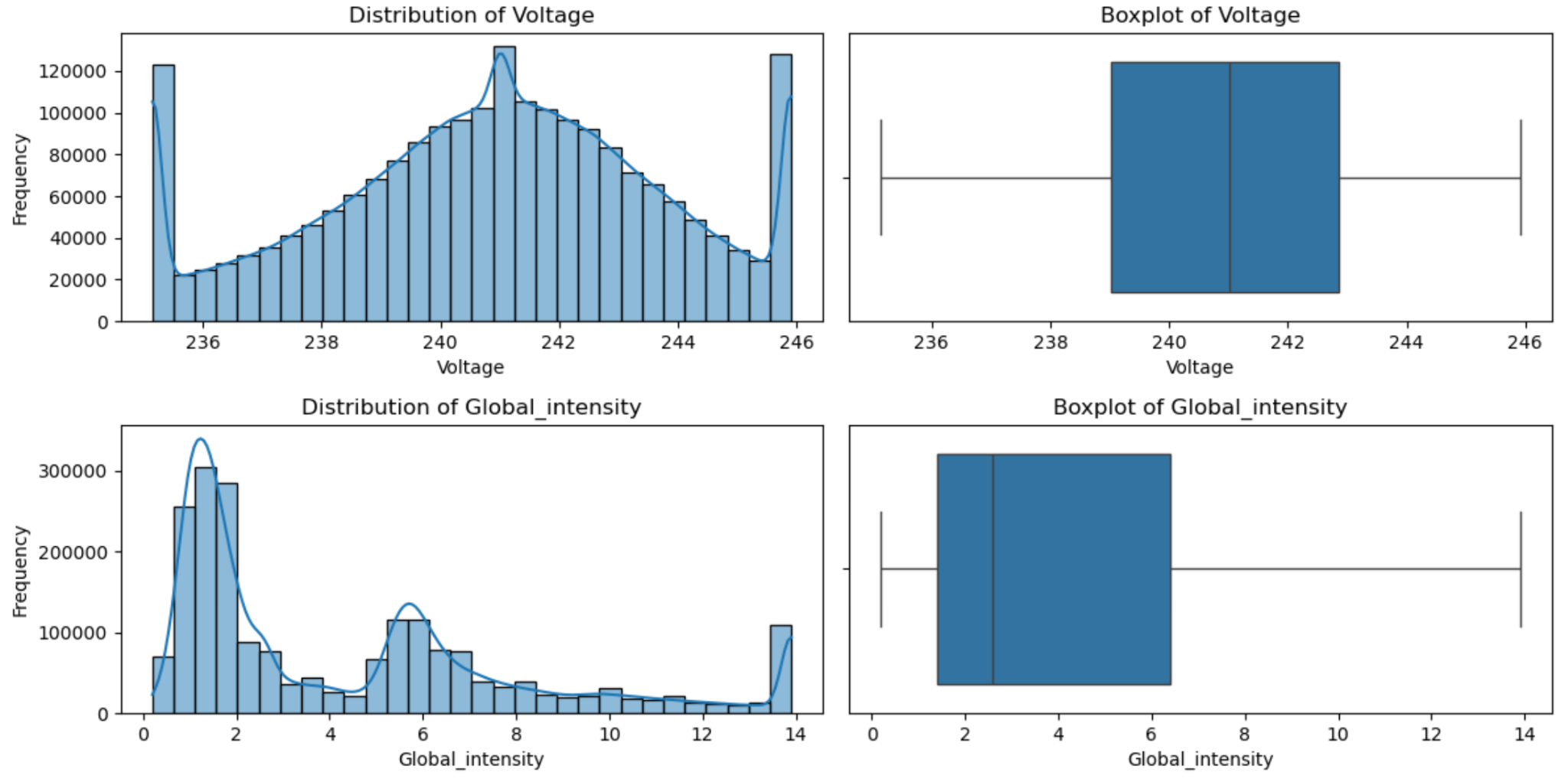
* For the **right-skewed** features—**Global\_active\_power**, **Global\_reactive\_power**, **Global\_intensity** and **Sub\_metering\_2**—the **IQR method** was applied. The IQR method is well-suited for right-skewed distributions, where outliers tend to be concentrated on the higher end. Outliers were clipped using this method, preserving the majority of the data while minimizing the effect of extreme values.
* For **Voltage**, which exhibited **both right and left skewness**, **Winsorization** was used. This method effectively handled outliers on both ends of the distribution, providing a balanced approach to manage extreme values without removing them entirely. Clipping the outliers in this feature ensured that the data remained comprehensive.
* For **Sub\_metering\_1**, which exhibited **right skewness**, the IQR method was initially attempted but was unable to effectively remove the outliers. As a result, **Winsorization** was applied to manage the outliers on the higher end. This allowed for a more practical handling of the extreme values while retaining the overall integrity of the data.

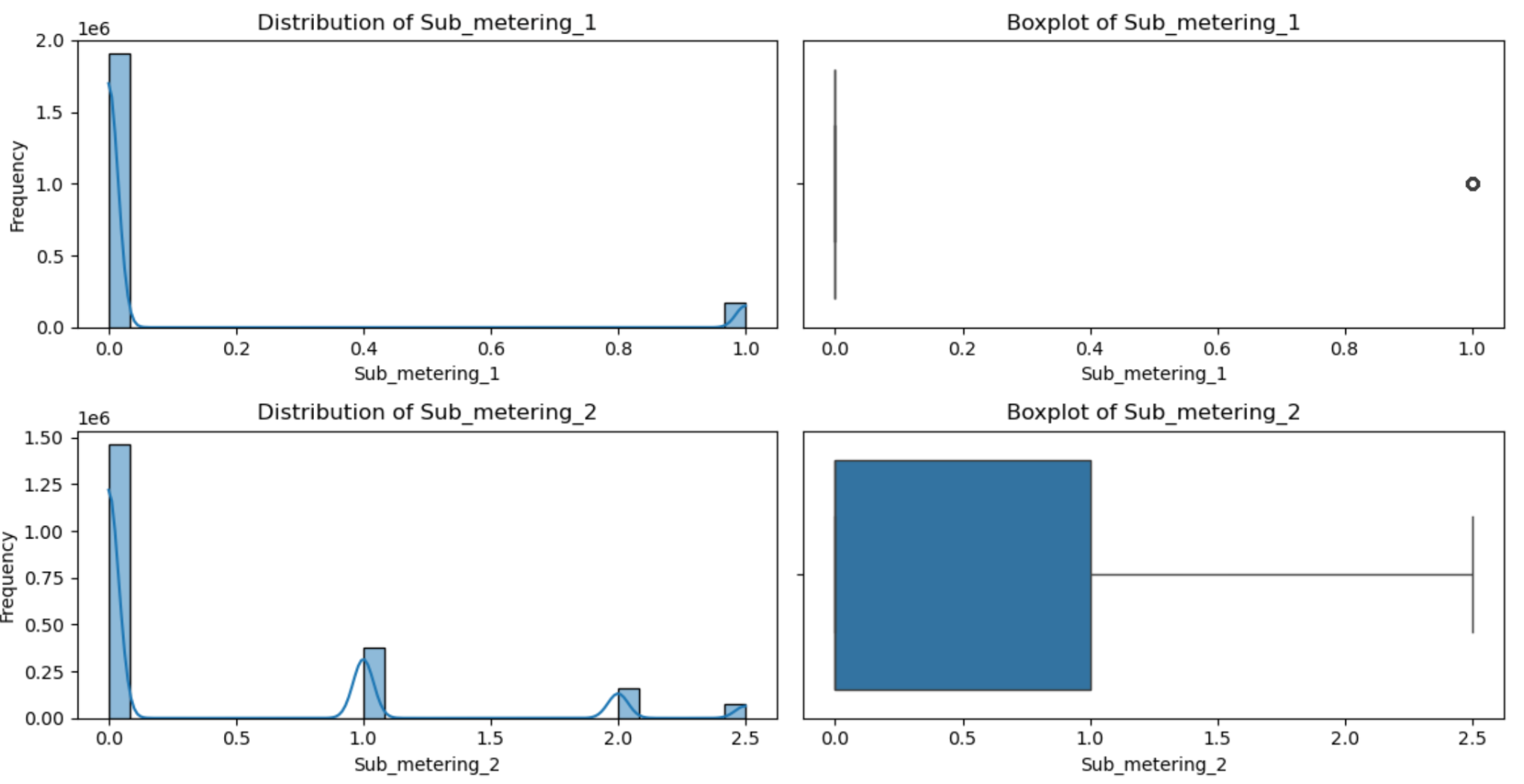
In both methods, **clipping** was performed instead of removing the outliers to avoid loss of data and ensure that valuable information was not discarded.

**Graphical Representation**

The below graphs represent the removal of outliers for the features which earlier had outliers.







**2. Feature Engineering**

To capture temporal patterns and gain deeper insights into energy consumption, several new features were engineered based on the existing Datetime column:

* Hour: Extracted the hour of the day to analyze hourly consumption trends.
* Day: Extracted the day of the week to observe any weekday vs. weekend consumption patterns.
* Month: Extracted the month to examine any seasonal trends in energy usage.
* Daily Average: Calculated the average energy consumption per day to smooth out day-to-day fluctuations.
* Peak Hour: Identified the hour with the highest energy consumption in a day.
* Rolling Average: Computed a rolling average to smooth short-term fluctuations and highlight longer-term trends.

After adding these features, the dataset was checked for missing values, and no missing values were found in any of the newly created features. Additionally, the data types of the new features were verified and found to be appropriate for further analysis and modeling.

**2.1 Handling Missing values:**

After adding these features, the dataset was checked for missing values, and no missing values were found in any of the newly created features. Additionally, the data types of the new features were verified and found to be appropriate for further analysis and modeling.

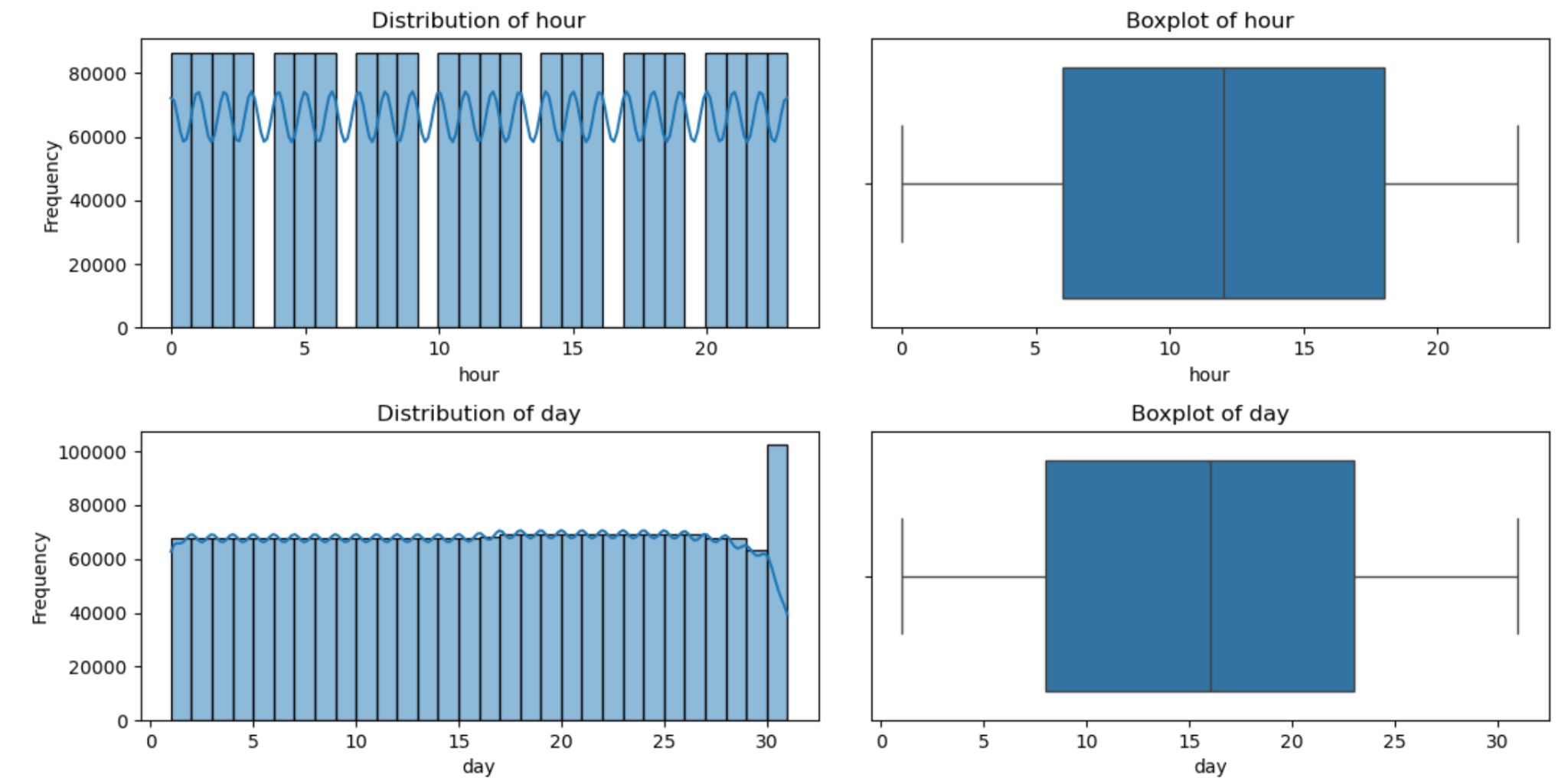
**2.2 Outlier Detection:**

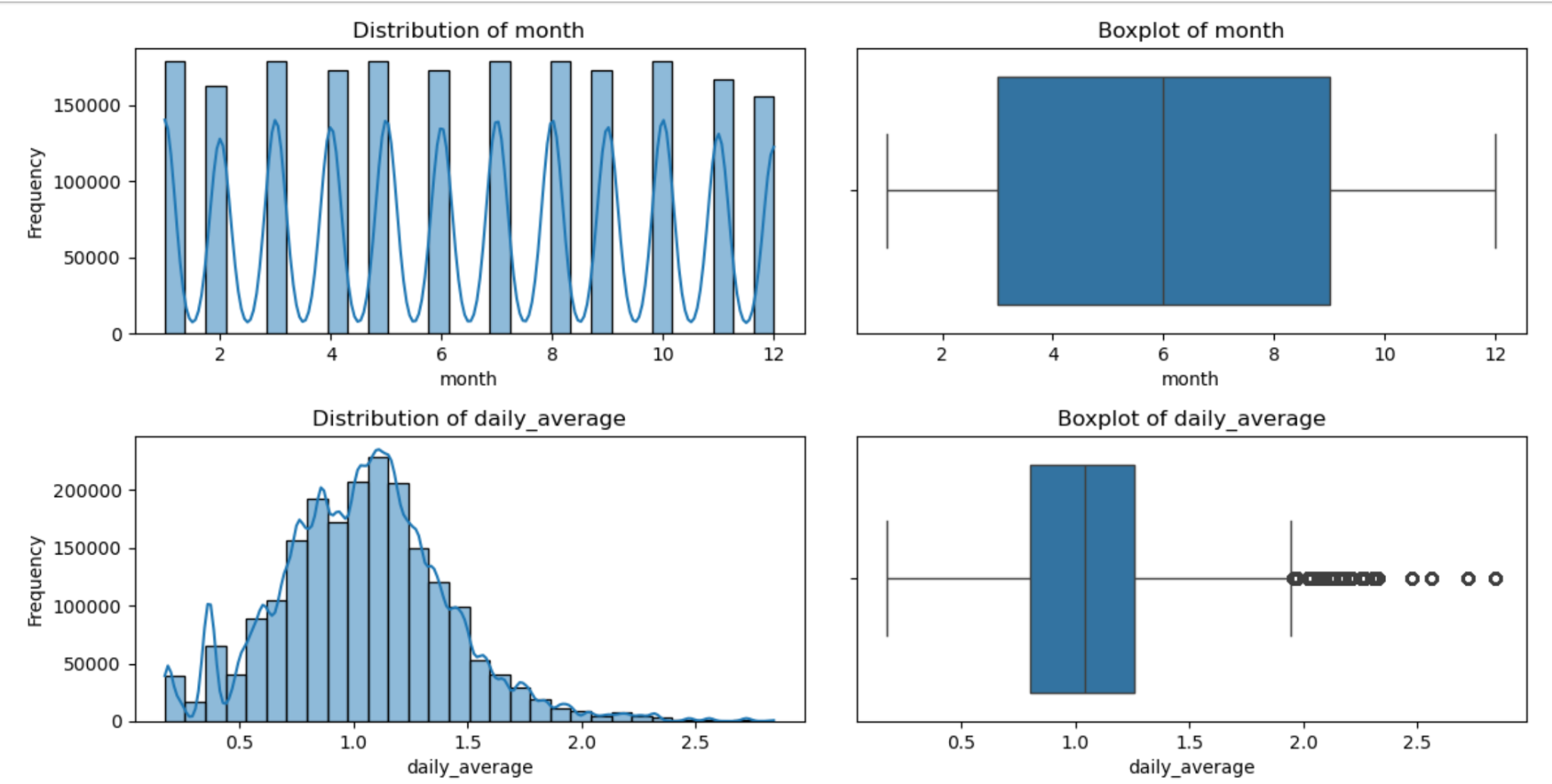
After handling missing values, outliers in the dataset were detected by visualizing the data using **boxplots** and **histograms**.

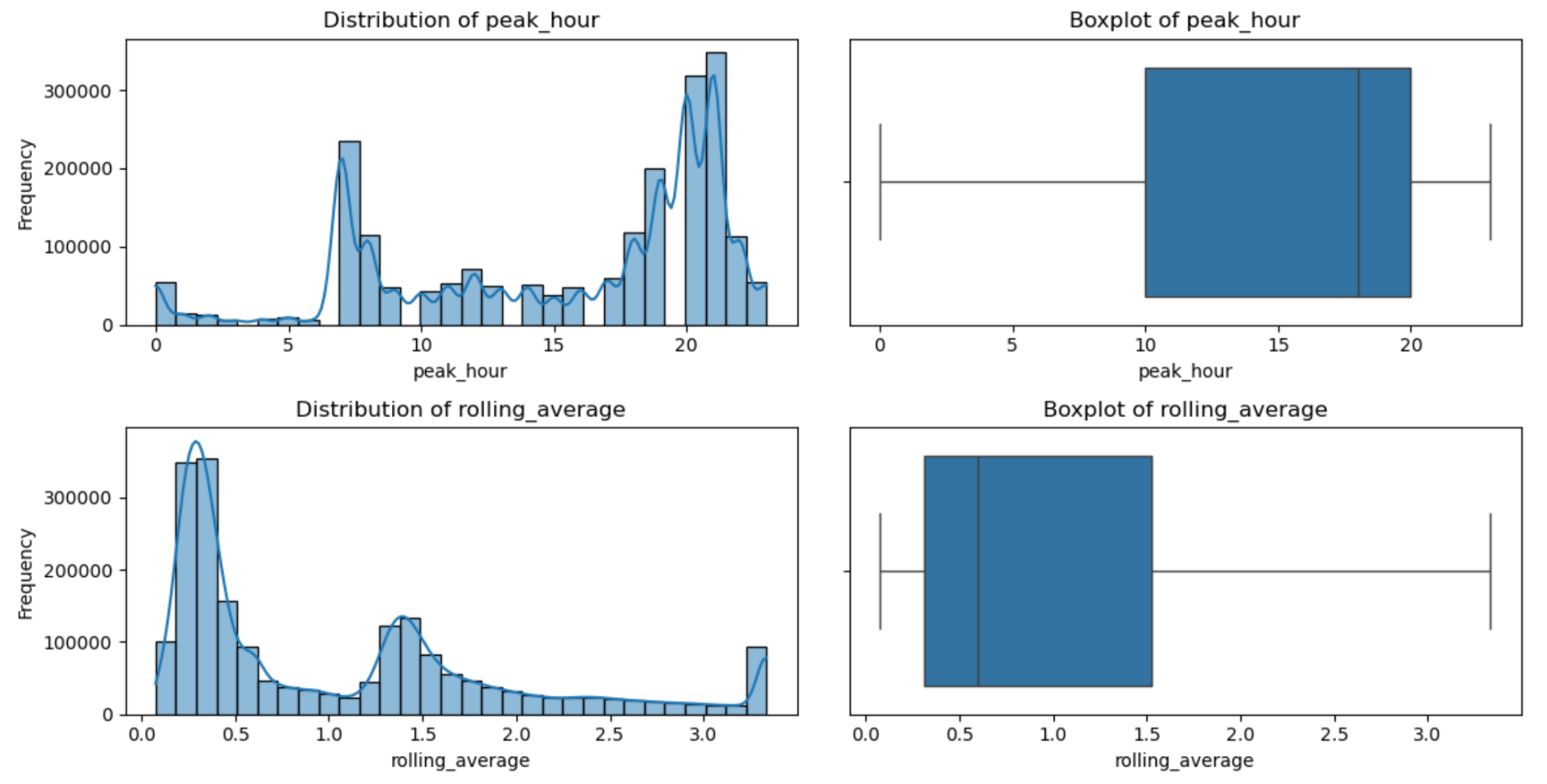
**Daily\_average** exhibited right-skewed distributions, indicating the presence of outliers on the higher end of the data.

**Graphical Representation:**

Below is the graphical representation to check for outliers in the newly added features

****

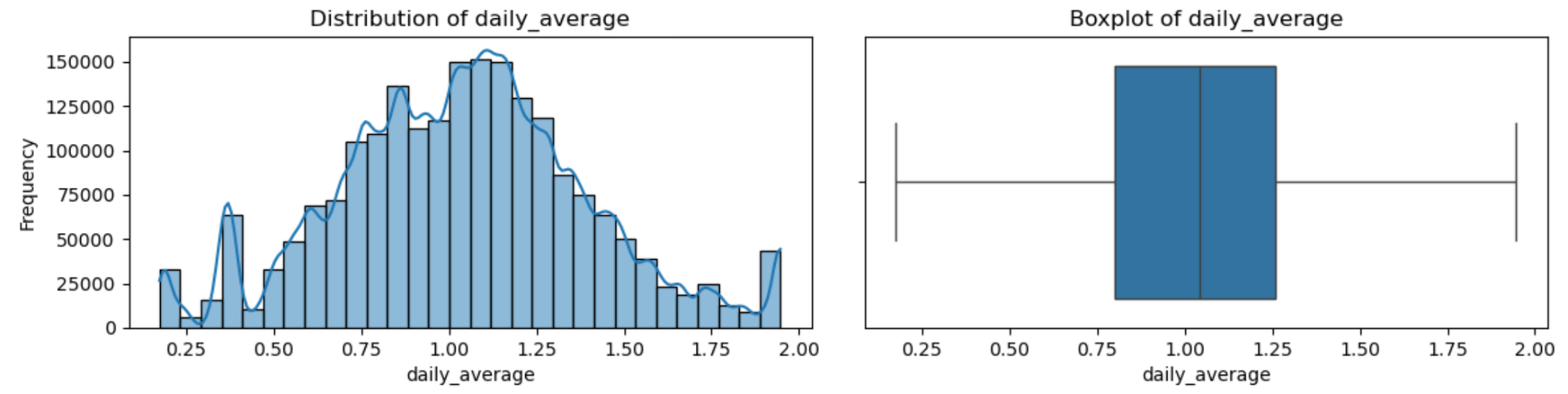
****

****

**2.3 Outlier Treatment:**

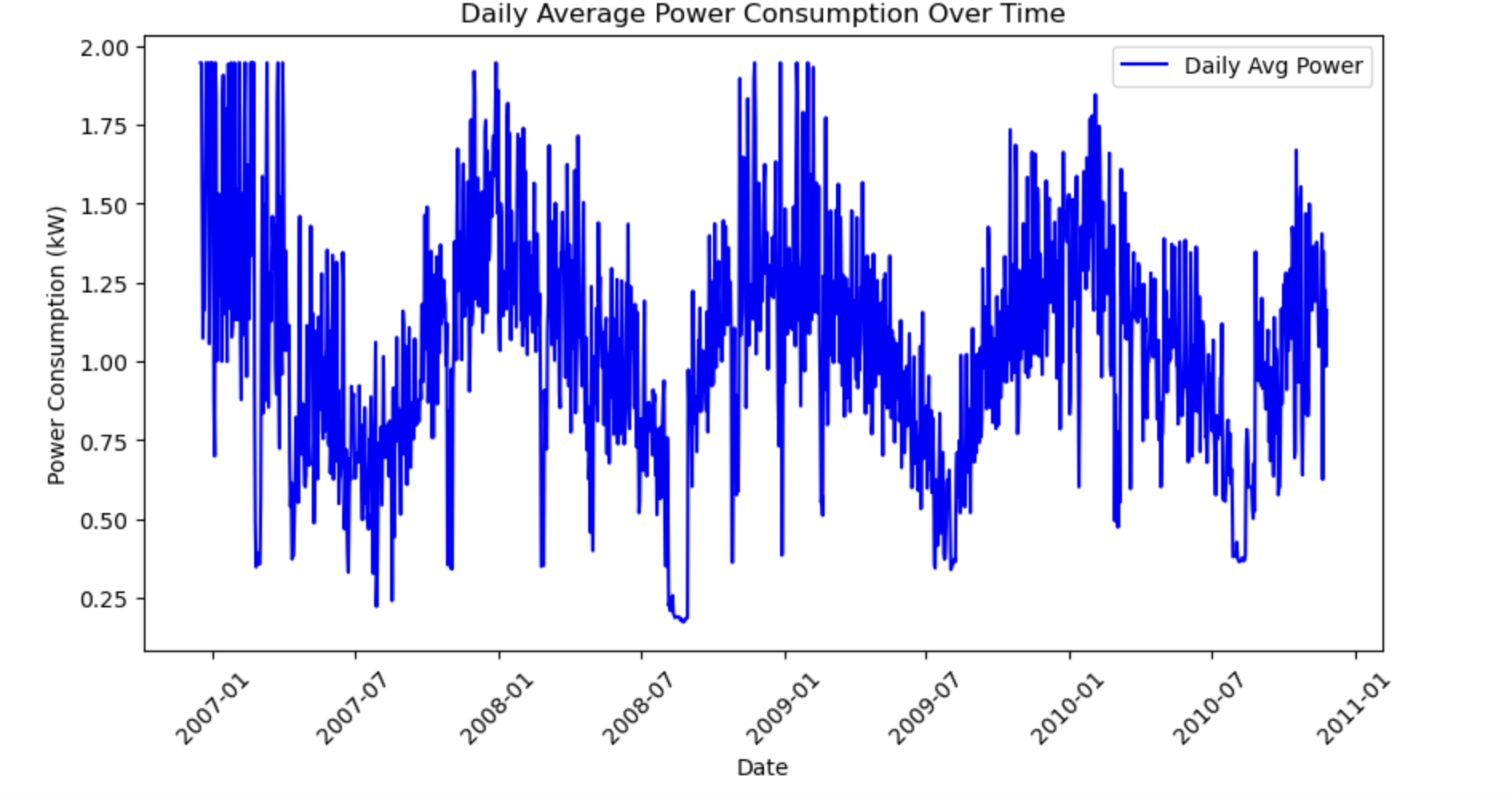
Only *daily\_average* had outliers, We applied the Interquartile Range (IQR) method to remove them. This feature exhibited a right-skewed distribution, and removing the outliers helped improve data quality, ensuring that extreme values did not negatively impact the model's performance.

Below graph represents daily\_average after outlier removal



**3. Exploratory Data Analysis:**

**Trend Analysis:**

1. **Trend Over Time Graph:**

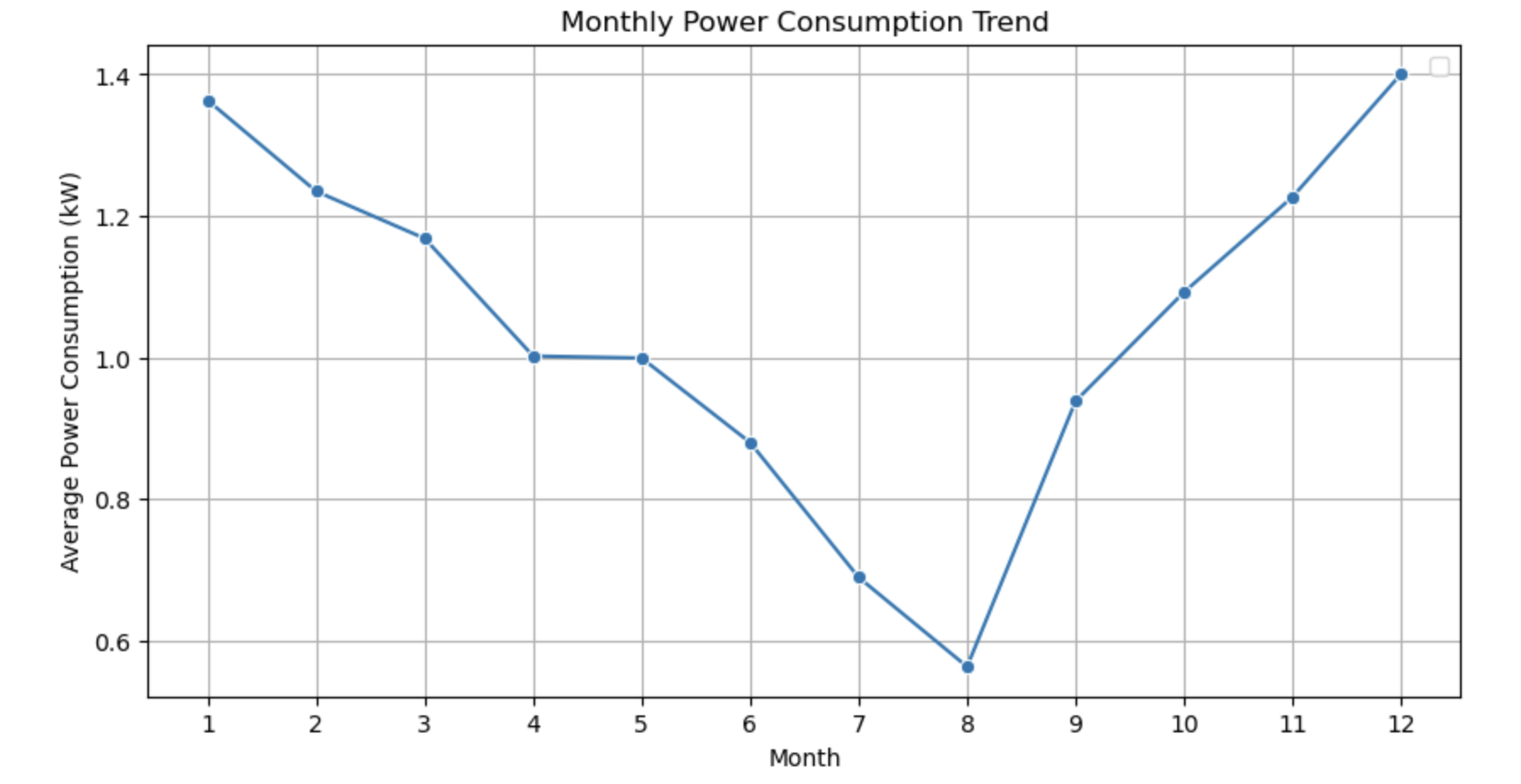
**Insights from the graph:**

Investigated the relationship between **Date** and **Power Consumption** to identify underlying patterns and trends.

A graph titled **"Daily Average Power Consumption Over Time"** was plotted, showing the variation in electricity usage from **January 2007 to January 2011**. Several key patterns were observed:

* **Fluctuating Usage**: Power consumption fluctuates significantly over time, exhibiting both high peaks and low troughs throughout the observed period. These fluctuations indicate variability in daily energy demands.
* **High Peaks and Seasonal Variations**:  
  The graph shows several instances where daily average power consumption reaches its peak near or above **1.75 kW**. These peaks may be linked to seasonal energy demands, such as increased heating during winter months or cooling during summer. This seasonal pattern results in higher average power consumption during these periods, reflecting the increased need for energy to maintain comfortable indoor environments.
* **Significant Drops**:  
  Occasional sharp drops in power consumption can be observed during certain months across all years. These drops may indicate periods of reduced overall energy demand, possibly during transitional seasons like spring or autumn when energy usage for heating or cooling is lower. Economic impacts or other external factors may also contribute to these declines in energy consumption.

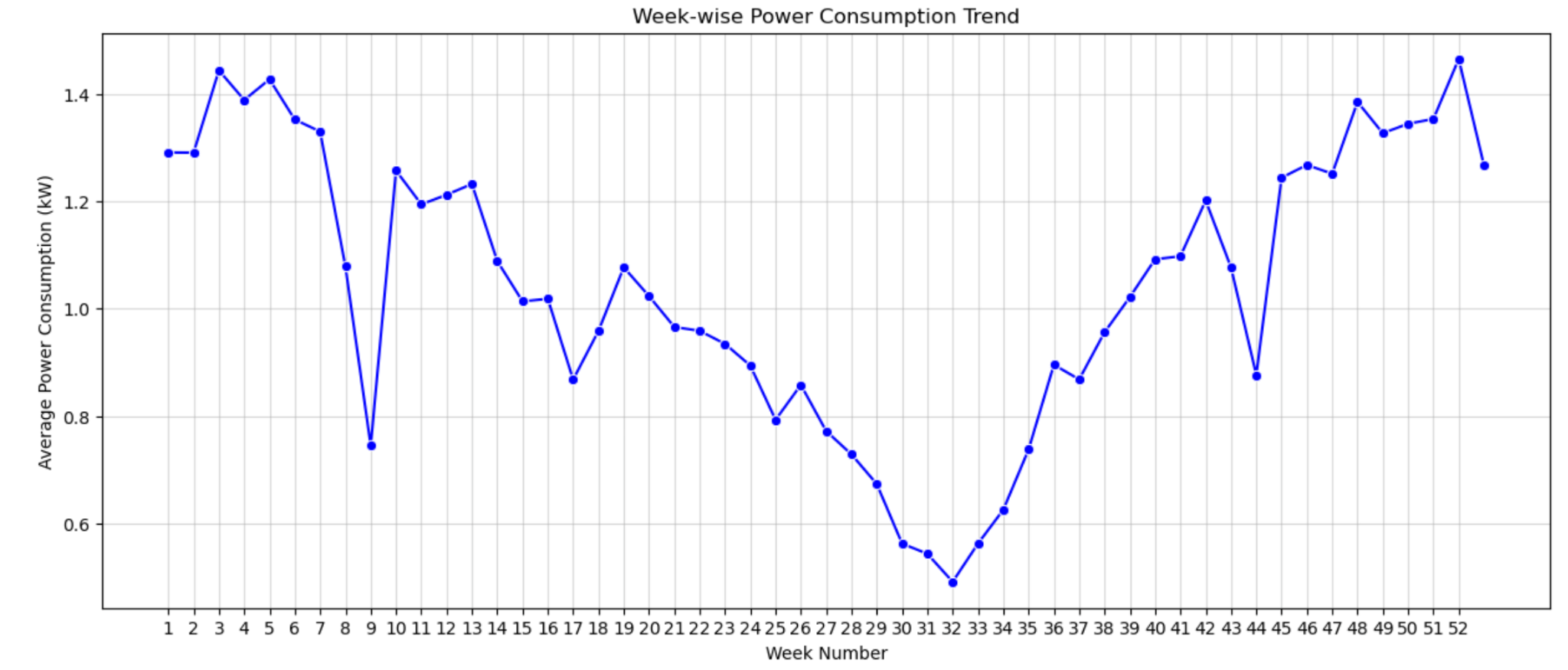
**2.Monthly Average Power Consumption Trend**

****

**Insights from the graph:**

* **Initial Peak:**The graph indicates the highest power consumption in the first month (likely January), with a value around 1.4 kW. January stands out as a period of high consumption, likely driven by heating needs during the winter months.
* **Declining Trend:**  
  From month 1 to month 8, there is a noticeable downward trend in power consumption, dropping significantly to around 0.6 kW mid-year. This reflects decreased energy demands, possibly due to milder weather conditions in the warmer part of the year.
* **Subsequent Increase:**  
  After month 8, power consumption gradually rises again towards the end of the year, particularly in the last quarter. This recovery phase in energy demand may be influenced by factors such as holiday seasons or other seasonal activities leading to increased energy use.

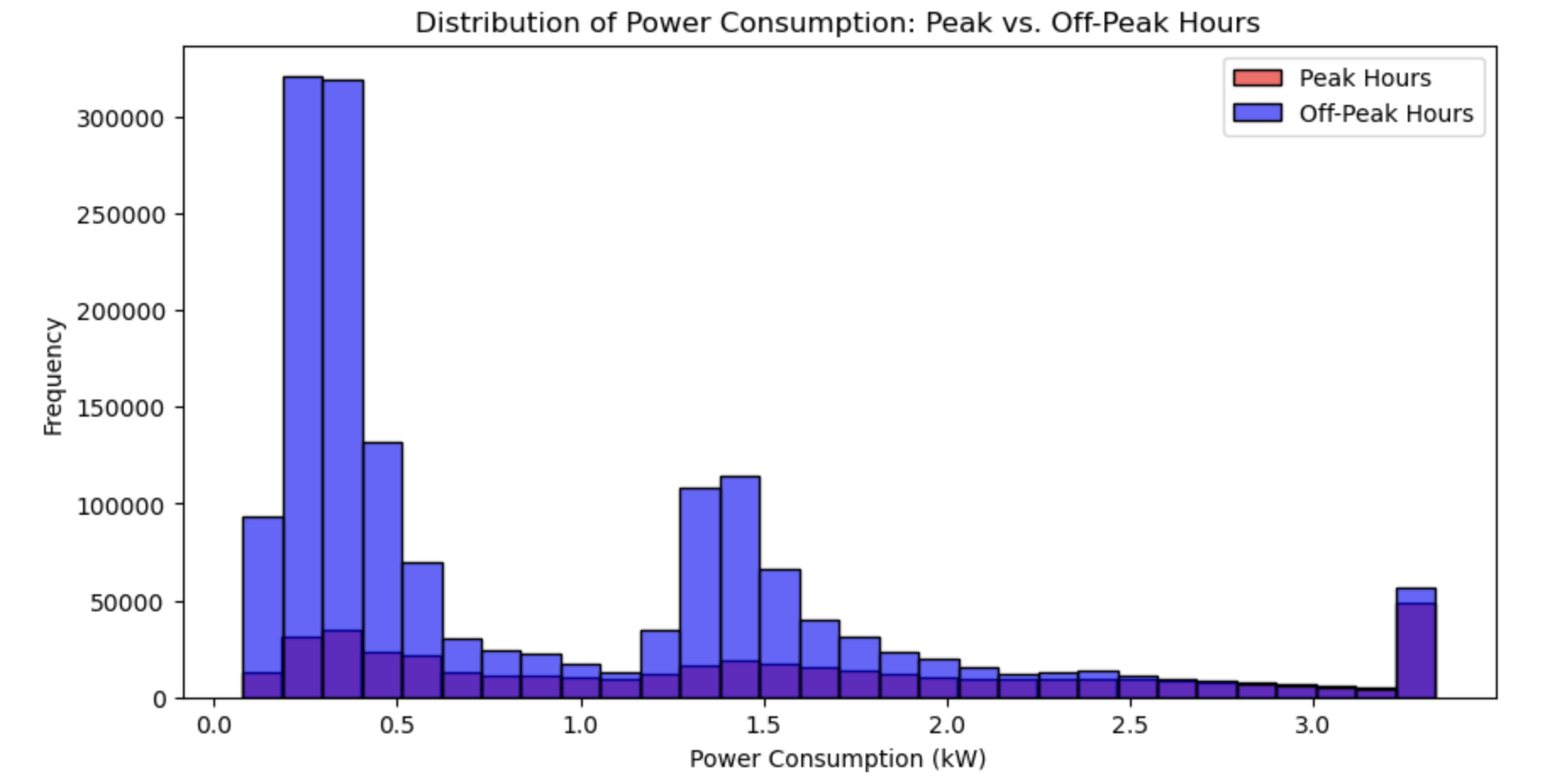
**3. Weekly Average Power Consumption**

****

**Insights from the graph:**

* **Peaks in Consumption:**Notable peaks are observed around Week 3, Week 5, and Week 52, where average power consumption exceeds 1.4 kW. These peaks may correspond to periods of increased energy use due to colder weather or specific seasonal activities.
* **Significant Drop:**A sharp decline in power consumption is evident around Week 9 and Week 32, with the lowest point dipping slightly below 0.8 kW. This suggests periods of reduced energy demand, possibly linked to milder weather conditions or lower overall activity.
* **Subsequent Increase:**Following the decline, there is a gradual rise in average power consumption, peaking again around Week 19 before showing some variation in the subsequent weeks.
* **Upward Trend Towards the End of the Year:**From around Week 36, there is a noticeable increase in power consumption, marking an upward trend towards the end of the year. This trend continues, with additional peaks occurring around Week 48 and Week 52, potentially reflecting holiday season energy demands.

**4. Peak vs off Peak Power consumption**

****

**Insights from the graph:**

* Highest frequency of power consumption is around 0.0 - 0.5 kW.
* Off-peak hours show higher frequency across most power consumption bins.
* Peak consumption values are lower in frequency compared to off-peak hours in most ranges.

**4. ML Model Selection, Training and Evaluation:**

In this phase, we explored several machine learning models to predict household energy consumption. The goal was to identify the model that best captures the relationship between the features and the target variable. We experimented with multiple models, including Linear Regression, Random Forest Regressor, Gradient Boosting Regressor, and a Neural Network to compare their performance.

For the Linear Regression model, we tested for multicollinearity using the Variance Inflation Factor (VIF) to ensure that the features were independent of each other and to improve the interpretability of the results.

To improve the generalization of the models and reduce overfitting, we performed hyperparameter tuning using techniques such as grid search and cross-validation. This process helped optimize the models’ performance and avoid overfitting.

The process involved the following key steps:

* **Model Selection:** Evaluating different models based on their ability to handle the dataset’s characteristics and complexity.
* **Training:** Training the models on the dataset to learn from historical data and discover the patterns in energy consumption.
* **Hyperparameter Tuning:** Fine-tuning the model parameters to enhance performance and minimize overfitting.
* **Evaluation:** Assessing the models’ performance using evaluation metrics like RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), and R² score to determine the most accurate and reliable model for predictions.

**4.1** **Linear Regression Model**

Linear Regression was the first model we employed to predict household energy consumption. This model was chosen for its simplicity and ease of interpretation, allowing us to explore linear relationships between the input features and the target variable.

**Multicollinearity Test:**

Before training the model, we conducted a test for multicollinearity using the Variance Inflation Factor (VIF). Initially, some features like **Global\_intensity,rolling\_average** and **Voltage** had very high VIF values, indicating severe multicollinearity. To address this, we iteratively removed the features with the highest VIF values. First, **Global\_intensity** was removed, which reduced multicollinearity. However, **Voltage** still showed a high VIF, so it was also removed. After these adjustments, the VIF values for the remaining features were within acceptable limits, ensuring that multicollinearity would not adversely affect the model's performance or the stability of the regression coefficients.

**Data Scaling:**

After addressing multicollinearity, we proceeded to train the linear regression model followed by scaling. We scaled the input features using **StandardScaler**. This step was essential to ensure all features had a mean of 0 and a standard deviation of 1, as Linear Regression can be sensitive to the scale of the input features. The scaler was fitted on the training data and subsequently applied to both the training and testing datasets for consistency.

**Model Evaluation:**

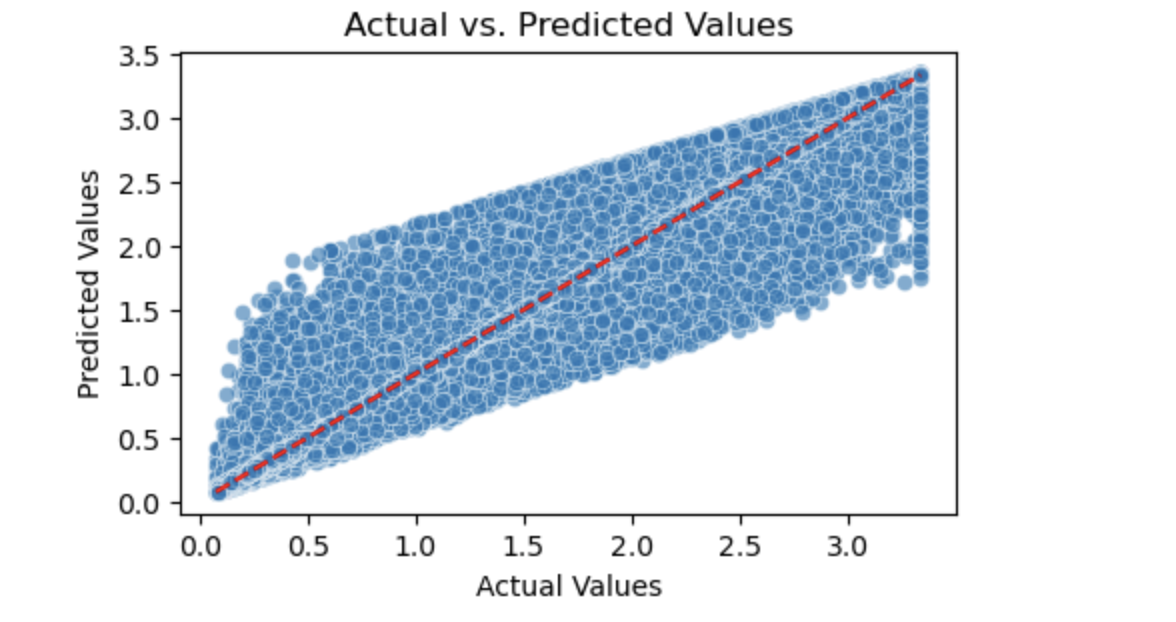
* **Root Mean Squared Error (RMSE):** To evaluate the average magnitude of prediction errors.
* **Mean Squared Error (MAE):** To measure the absolute squared difference between predicted and actual values.
* **R² Score:** To determine the proportion of variance in the target variable explained by the model.

Despite not performing hyperparameter tuning, the linear regression model yielded the following results:

* **Training RMSE:** 0.11
* **Training MAE:** 0.03
* **Training R² Score:** 0.99
* **Testing RMSE:** 0.11
* **Testing MAE:** 0.03
* **Testing R² Score:** 0.99

The high R² score and low error metrics indicate that the linear regression model demonstrated excellent performance on both the training and testing datasets. The close alignment between the training and testing errors suggests that the model has effectively learned the underlying patterns in the data without overfitting. There is no significant performance degradation on the testing set, which confirms that the model generalizes well to unseen data. These results establish a robust foundation and offer a strong baseline for further experimentation with more complex and advanced models.

Below is the Graphical Representation between Actual values and the Predicted Values



**4.2 Random Forest Regressor**

After the linear regression model, we employed the Random Forest Regressor, an ensemble learning method known for its ability to handle non-linear relationships and prevent overfitting in many cases.

#### **Initial Model Performance**

We first trained and tested the model with the default hyperparameters, yielding the following results:

* Training RMSE: 0.01
* Training MAE: 0.00
* Training R² Score: 1.0
* Testing RMSE: 0.02
* Testing MAE: 0.00
* Testing R² Score: 1.0

##### **Insights:**

1. **Overfitting:** The model showed a perfect fit on the training data, with an R² score of 1.0 and almost zero error. However, such results are often a sign of overfitting, meaning the model is learning the training data too well, capturing noise and specific patterns that do not generalize well to new data.
2. **Good Testing Performance:** While the testing data still showed excellent results with an R² score of 1.0, the small difference between training and testing RMSE suggests that the model may not generalize as well as expected, especially on unseen data from different distributions. This indicates potential overfitting despite the high accuracy on the testing set.

Given these insights, we decided to fine-tune the hyperparameters to reduce overfitting and improve the model's ability to generalize.

### 

### **Hyperparameter Tuning**

### To address the overfitting observed in the initial Random Forest model, we performed hyperparameter tuning using RandomizedSearchCV to improve its generalization ability. Given the large size of the dataset, we encountered kernel crashes during the tuning process. As a result, we used more conservative values for the hyperparameters and limited the number of iterations and cross-validation folds to prevent excessive computational load.

After running the tuning process, the model performance metrics improved:

* Training RMSE: 0.16
* Training MAE: 0.12
* Training R² Score: 0.97
* Testing RMSE: 0.16
* Testing MAE: 0.12
* Testing R² Score: 0.97

##### **Insights:**

1. **Reduced Overfitting:**

After tuning, the model's performance became more balanced between the training and testing datasets. The slight increase in training error (RMSE of 0.16) compared to the initial model indicates that the model is now better regularized and less overfit to the training data. The testing performance remains strong, with both RMSE and R² scores of 0.97 for the training and testing datasets, indicating improved generalization.

1. **Better Generalization:**

The tuned model no longer exhibits extreme overfitting and performs consistently across both the training and testing datasets. This consistency suggests that the hyperparameter adjustments, such as limiting tree depth and increasing the minimum number of samples for splits and leaves, have effectively reduced overfitting and enhanced the model's robustness. As a result, the model is now more capable of generalizing to new, unseen data.

**4.3 Gradient Boosting Regressor Model**

We next applied the Gradient Boosting Regressor, a powerful ensemble technique that builds multiple trees sequentially, with each tree correcting the errors of the previous ones.

#### **Initial Model Performance:**

* Training RMSE: 0.03
* Training MAE: 0.02
* Training R² Score: 1.00
* Testing RMSE: 0.03
* Testing MAE: 0.00
* Testing R² Score: 1.00

#### **Insights:**

* **Overfitting Concerns:** The initial model results suggest potential overfitting. Both training and testing metrics are nearly perfect, with an R² score of 1.0 and almost zero error. While this can indicate an excellent fit to the data, such near-perfect scores raise concerns that the model may be learning the noise and specific patterns of the training data rather than generalizing to new, unseen data.
* **Perfect Fit:** Although the testing data also shows impressive results, the absence of any error indicates that the model could have memorized the training data, which may cause performance degradation on different testing datasets. Given these signs, we decided to perform hyperparameter tuning to reduce overfitting and improve generalization.

### **Hyperparameter-Tuned Gradient Boosting Model**

To address the overfitting observed in the initial Gradient Boosting model, we conducted hyperparameter tuning using RandomizedSearchCV. This process allowed us to find optimal parameter values that could strike a balance between model complexity and performance.

#### **Tuned Model Performance:**

* Training RMSE: 0.07
* Training MAE: 0.04
* Training R² Score: 0.99
* Testing RMSE: 0.07
* Testing MAE: 0.04
* Testing R² Score: 0.99

#### **Insights:**

**Reduced Overfitting:** After tuning, the slight increase in training error indicates that the model is no longer overfitting, as it avoids perfectly capturing the training data's noise and patterns.

**Improved Generalization:** The consistent performance between training and testing after tuning shows that the model now generalizes better to new data, maintaining high accuracy while being more resilient to overfitting. This reduces the likelihood of performance drop on unseen datasets.

**4.4 Neural Network Model**

After exploring traditional machine learning models, we applied a neural network using TensorFlow and Keras to predict household energy consumption. Neural networks are well-suited for capturing complex, non-linear relationships between features, making them a valuable approach for this regression task.

#### **Model Architecture:**

* **Input Layer:** The input layer receives the scaled features. We normalized the data using StandardScaler to improve the model’s performance, as neural networks generally work better with scaled inputs.
* **Hidden Layers:**
  + The first hidden layer consists of 128 neurons with the ReLU activation function.
  + The second hidden layer contains 64 neurons, also using the ReLU activation.
* **Output Layer:** The output layer consists of 1 neuron without any activation function (i.e., linear activation), as we are performing a regression task.

The model is compiled using the Adam optimizer, and the mean squared error (MSE) is used as the loss function.

#### **Activation Function (ReLU):**

* We used the ReLU (Rectified Linear Unit) activation function in the hidden layers. ReLU is one of the most widely used activation functions in deep learning due to its simplicity and effectiveness. It allows the model to:
  + **Prevent the vanishing gradient problem:** Unlike sigmoid or tanh, which squash inputs to a small range, ReLU allows gradients to flow when inputs are positive, leading to faster and more efficient training.
  + **Increase computational efficiency:** ReLU is computationally cheaper than other activation functions, allowing for quicker training times.
  + **Model complex patterns:** ReLU helps the network learn non-linear relationships between the input features and the target, making it ideal for complex tasks like energy consumption prediction.

#### **Optimizer (Adam):**

* We used the Adam optimizer because it is highly effective for training deep learning models.
  + **Adaptively adjusts learning rates:** Adam adjusts the learning rate for each parameter, which helps the model converge faster and more efficiently without extensive manual tuning.
  + **Handles sparse gradients well:** Adam performs well on problems with noisy or sparse gradients, which can occur in complex datasets.
  + **Robust and computationally efficient:** Adam is generally robust to noisy data and gradients, and its computational efficiency makes it suitable for large datasets like ours.

#### **Training Process:**

The model was trained for 50 epochs with a batch size of 32. We used 20% of the training data for validation during training to monitor the model’s performance and prevent overfitting.

### **Model Evaluation:**

After training the model, we evaluated it on the test dataset and calculated various performance metrics.

#### **Model Performance:**

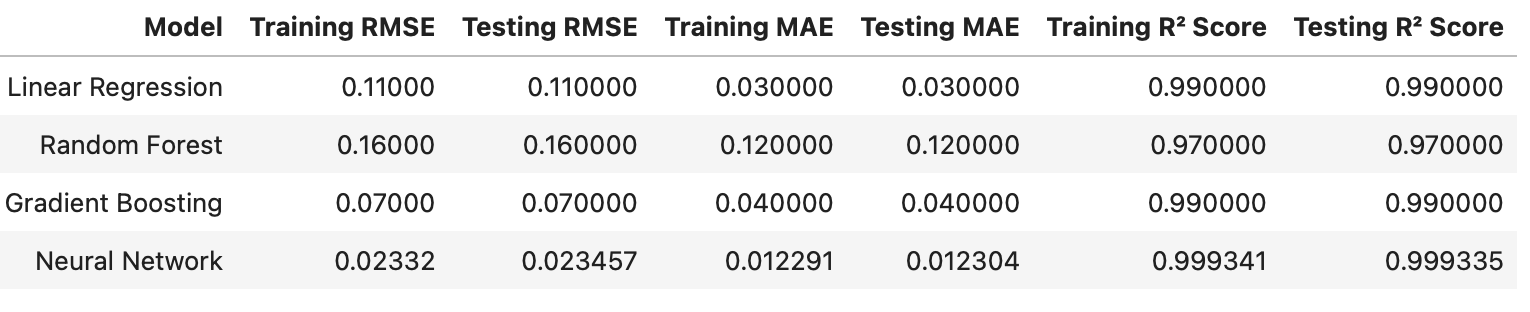
* Test RMSE: 0.02
* Test MAE: 0.0123
* Test R² Score: 0.9993
* Training RMSE: 0.02
* Training MAE: 0.0123
* Training R² Score: 0.9993

#### **Insights:**

* **High Accuracy:** The model performs exceptionally well, achieving nearly identical performance on both training and testing datasets, with an R² score of 0.9993 for both. This indicates that the model fits the data very well and captures the underlying patterns.
* **No Overfitting:** The minimal difference between training and testing RMSE and MAE suggests that the model generalizes well to new data without overfitting.
* **Low Error:** Both the training and testing RMSE values of 0.02 demonstrate that the model's predictions are highly accurate, with minimal error in predicting energy consumption.

**Model Comparison:**

Below table shows the performance of models in terms of RMSE, MSE, R2 Score

****

**Insights:**

1. **Neural Network**: The Neural Network model outperforms the other models with the lowest RMSE (0.023 for both training and testing) and MAE (0.012), indicating the smallest prediction errors. Its near-perfect R² score (0.9993) suggests that the model explains almost all the variance in the data, making it the most accurate and best-performing model in terms of predictive power and generalization. This model is highly recommended for its precision.
2. **Gradient Boosting**: Gradient Boosting ranks second with an RMSE of 0.07 for both training and testing, and a high R² score of 0.99. It also has relatively low MAE (0.04), demonstrating that it performs well with minimal prediction errors. While it's not as accurate as the Neural Network, it's still a very strong and reliable model. Gradient Boosting can also offer better interpretability, depending on the use case.
3. **Linear Regression**: Linear Regression comes in third with an RMSE of 0.11 and a solid R² score of 0.99. While it shows good overall performance and simplicity, its prediction errors are larger compared to the Neural Network and Gradient Boosting models. However, it is still a viable choice when model interpretability is more important, as it provides a straightforward linear relationship between features.
4. **Random Forest**: Random Forest ranks lowest with the highest RMSE (0.16) and MAE (0.12), along with the lowest R² score (0.97). This suggests that it has relatively larger prediction errors and doesn't capture the variance in the data as effectively as the other models. While it is robust and can handle noisy data or outliers well, it is less accurate in this case compared to the other models.

**Conclusion**: Based on the performance metrics (RMSE, MAE, and R²), the **Neural Network** is the best model for accuracy and generalization, followed by **Gradient Boosting**. **Linear Regression** is simpler but less precise, and **Random Forest** lags behind in this particular scenario.