Data Analysis & Visualization



Project title: Occupancy Detection using PIR Sensors

Group Members

Samreen Fatima Kazmi

Maryam Noor Ul Ain

Laraib Sultana

Instructor: Prof M.Bilal Haidral

Namal University Mianwali





### 

Table of Contents

[**Abstract:**](about:blank) 1

[2](#_u16qqrpfboyq)

[**Acknowledgments:**](about:blank) 2

[**Overview:** 5](#_4z5613hyhqy9)

[**EDA:** 5](#_kjxncds7jsjo)

[**Data Transformation:** 15](#_3k4z3vp46x1u)

[**Regression Models** 20](#_dvjr9o1rmt4e)

[**For Cleaned data:** 21](#_5otjp0jhn0mk)

[**Linear regression (Single Variable):** 22](#_albxyrm82sn2)

[**Linear regression (Multivariable):** 23](#_kdnn493b3s2j)

[**Polynomial Regression (Single Variable):** 24](#_ns0vnbx781fe)

[**Polynomial Regression (Multivariable):** 24](#_a7qu51472y23)

[**Decision Tree regressor:** 25](#_q6vzzh56g3io)

[**Comparative Model Performance Ranking** 26](#_u1975wdm52if)

[**For Scaled data:** 26](#_5f7v5mo594d)

[**Linear regression (Single Variable):** 27](#_d4w6urt10c5r)

[**Linear regression (Multivariable):** 28](#_2m69h9vn1ssv)

[**Polynomial Regression (Single Variable):** 29](#_mg16rmi8mklg)

[**Polynomial Regression (Multivariable):** 29](#_x0c6h980dk4b)

[**Decision Tree regressor:** 30](#_rg2ikjhhmkor)

[**Comparative Model Performance Ranking** 31](#_tnoa19ddsjpd)

[**Comparative Analysis: Cleaned vs Scaled Data Performance** 31](#_25a4pacmyg9t)

[**Key Insights from the Comparison** 31](#_u9caplt680sp)

[**Classification models on cleaned data:** 32](#_b48jm2ibij7c)

[**Data preparation:** 32](#_kj68d1iud3ol)

[**Data Splitting:** 32](#_i8uq31jmdnpa)

[**Model Training:** 32](#_riwnv15oy4q6)

[**Model Evaluation:** 33](#_ffpv60di64f6)

[**Visualization:** 33](#_jnhjy8bd16u0)

[**Classification models on Scaled data:** 36](#_w9tdmos9m9mj)

[**Data Preparation:** 36](#_pirp50f5mm9o)

[**Data Splitting:** 36](#_o9658iytksr2)

[**Model training:** 37](#_dj8efsetqr5m)

[**Model Evaluation:** 37](#_vazw8l11su4o)

[**Visualization:** 37](#_yw01hlj5f0ah)

[**Time Series Analysis:** 38](#_gnol24pmqlj1)

[**Auto Regression (AR) Model:** 39](#_28na9ghsizga)

[**Moving Average (MA) Model:** 40](#_r5rqegolii7v)

[**AutoRegressive Moving Average (ARMA) Model:** 42](#_amxvy4eek39m)

[**AutoRegressive Integrated Moving Average (ARIMA) Model:** 43](#_10rvdgjts6v9)

[**Seasonal ARIMA (SARIMA) Model:** 45](#_2mb1tmnzby9b)

[**Clustering Models on Scaled Data:** 46](#_p3qvc9ygjtlf)

[**Comparative Analysis** 48](#_1tmmtoqsh1ww)

[**Conclusion** 48](#_cvaacyrqi21j)

[**Research question** 49](#_tok5e3vwqys1)

[**Question 1:** 49](#_ozk9gow6u2w6)

[**Question 2:** 53](#_2ev10e11ynja)

[**Performance:** 53](#_dlj1c65yeq8u)

[**Static Baseline Model:** 53](#_rij0n9r72dcn)

[**LLE Model accuracy trend:** 54](#_yq3gcffkni7x)

[**Critical Analysis:** 54](#_2ce6agyowry6)

[**Strengths** 55](#_rd53p3lonr2g)

[**LLE Model Conclusion:** 55](#_3ezz269c0zyh)

[**Conclusion:** 55](#_ixg3glfc45qr)

## **Overview:**

This project aims to detect human presence in indoor environments using data collected from Passive Infrared (PIR) sensors. PIR sensors detect motion by measuring infrared radiation levels emitted by objects in their field of view. When a person enters or moves in a room, the sensor records activity, making it a reliable tool for occupancy detection.

**Name**: PIRvision\_FoG\_presence\_detection dataset

**Structure**: Timestamped readings from multiple PIR motion sensors

**Target Variable**: label with 3 classes (0, 1, 3)

**Features**: Raw sensor readings and engineered features

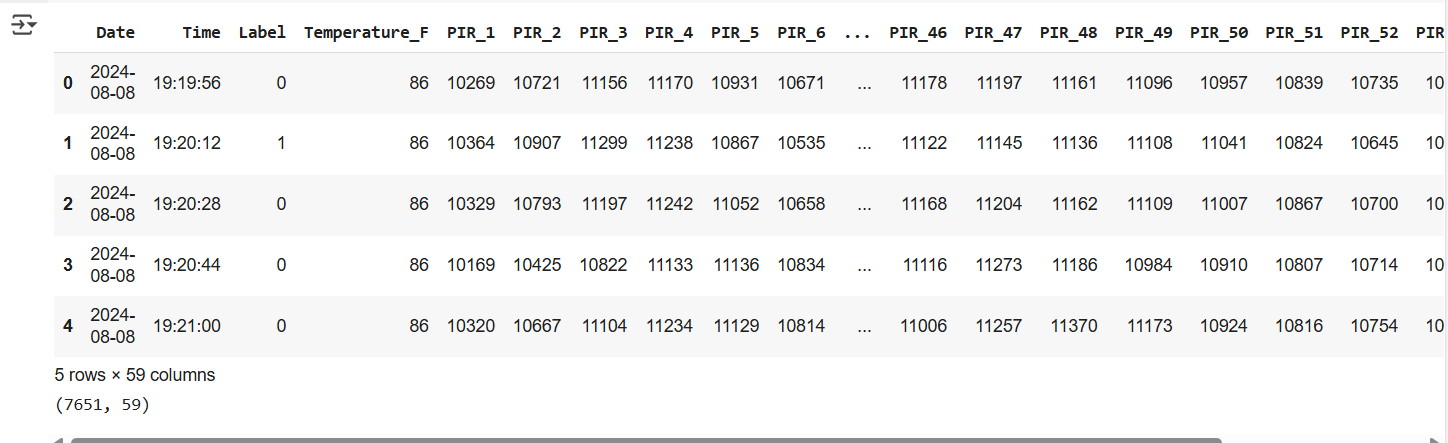
**Challenges**:

* **Multi-class Imbalance**: Class 3 has fewer samples
* **Sensor Noise**: Environmental IR sources can mimic human presence
* **Temporal Dynamics**: Sensor data must be aligned with human behavior over time

## **EDA:**

**1. Data Loading**

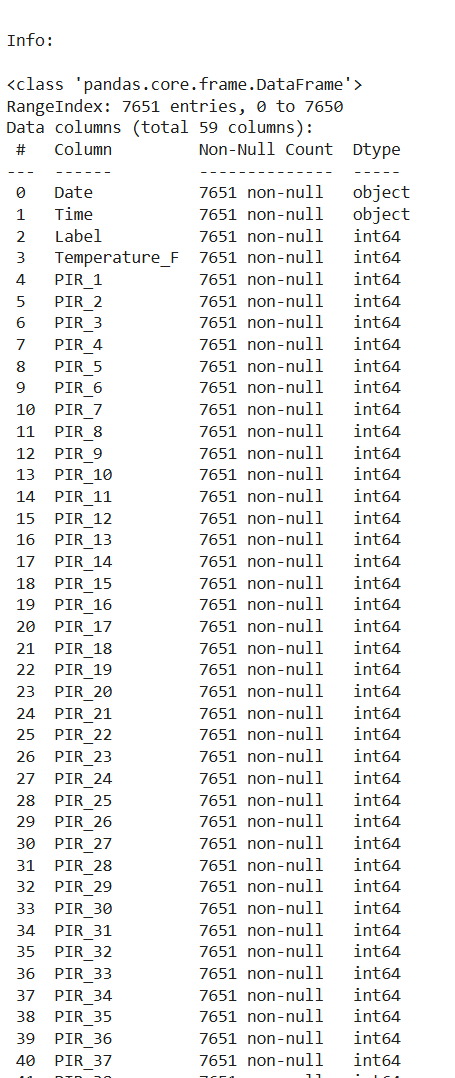
The dataset "pirvision\_office\_dataset1.csv" was loaded into a pandas DataFrame. The first few rows and the shape of the DataFrame were displayed to verify successful loading and get an initial understanding of the data structure.

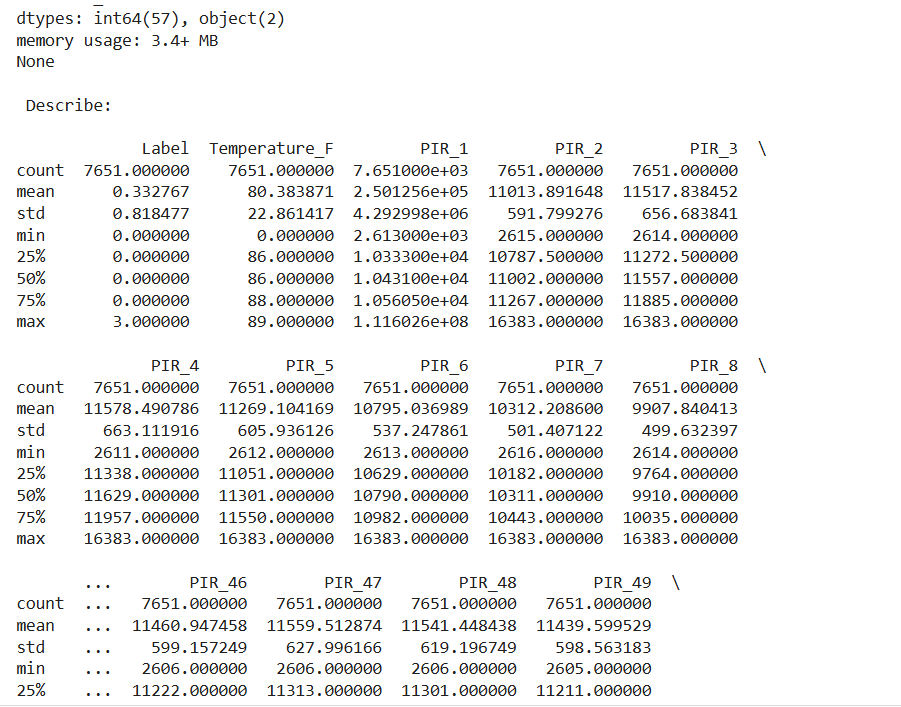


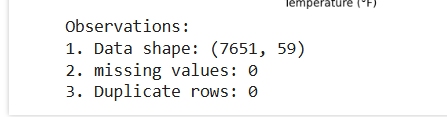
**2. Data Exploration**

The dataset was explored to understand its characteristics. This involved:

* Examining data types and summary statistics using df.info() and df.describe().
* Checking for duplicate rows using df.duplicated(). The number of duplicate rows found was reported.



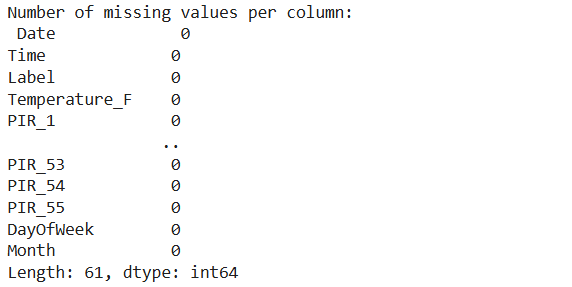




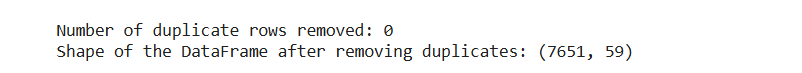
**3. Data Cleaning**

The data was cleaned by handling missing values, removing duplicate rows, and addressing outliers.

* **Missing Values:** The number of missing values per column was checked using df.isnull().sum(). No missing values were found in the dataset, so no action was taken.



* **Duplicate Rows:** Duplicate rows were removed using df.drop\_duplicates(). The number of duplicate rows removed and the shape of the DataFrame after removal were reported.

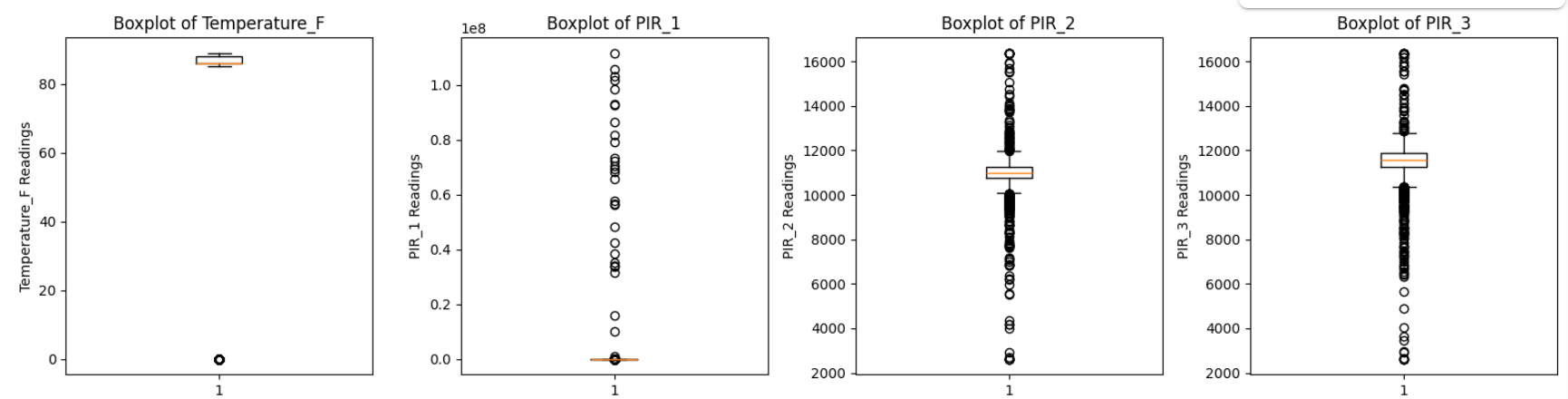


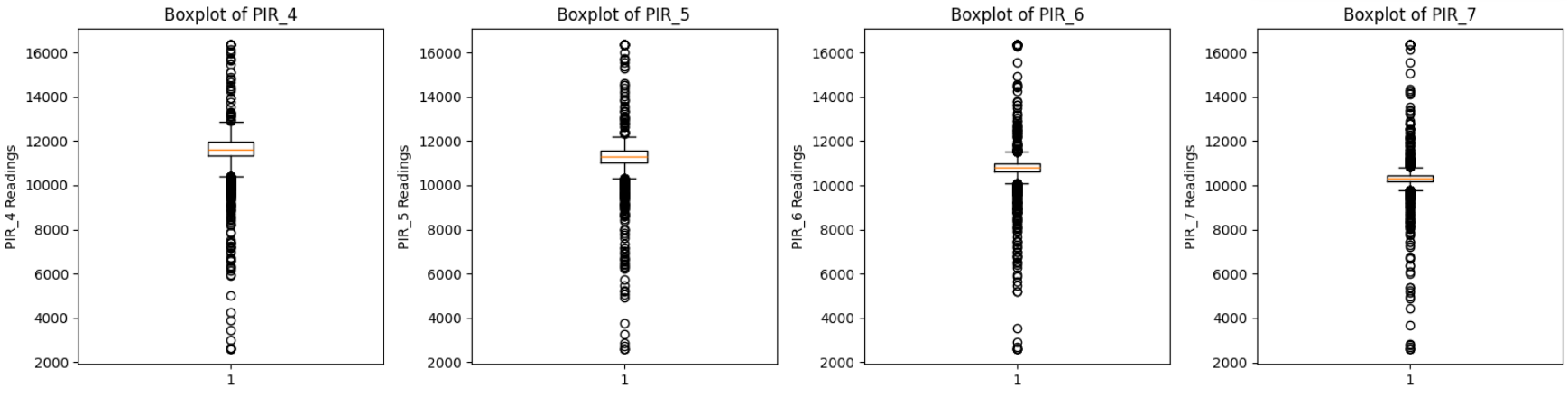
* **Outliers:** Outliers were removed by calculating the Interquartile Range (IQR) for each numerical feature. Data points falling below Q1 - 1.5\*IQR or above Q3 + 1.5\*IQR were filtered out of the dataset. This process was applied iteratively to each numerical column.

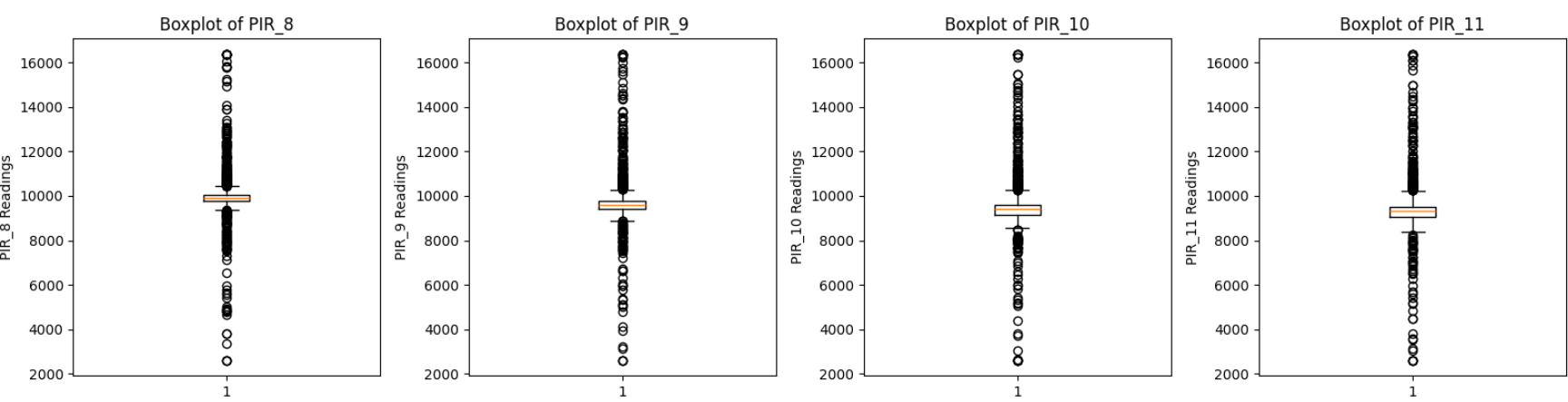


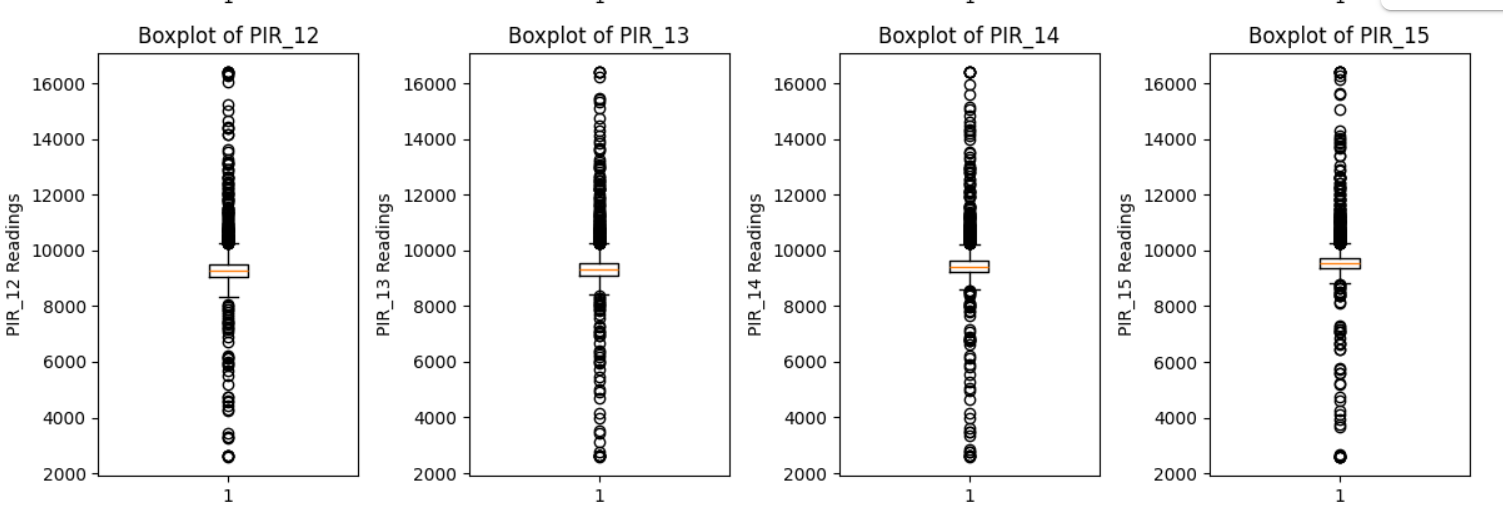
**4. Data Visualization**

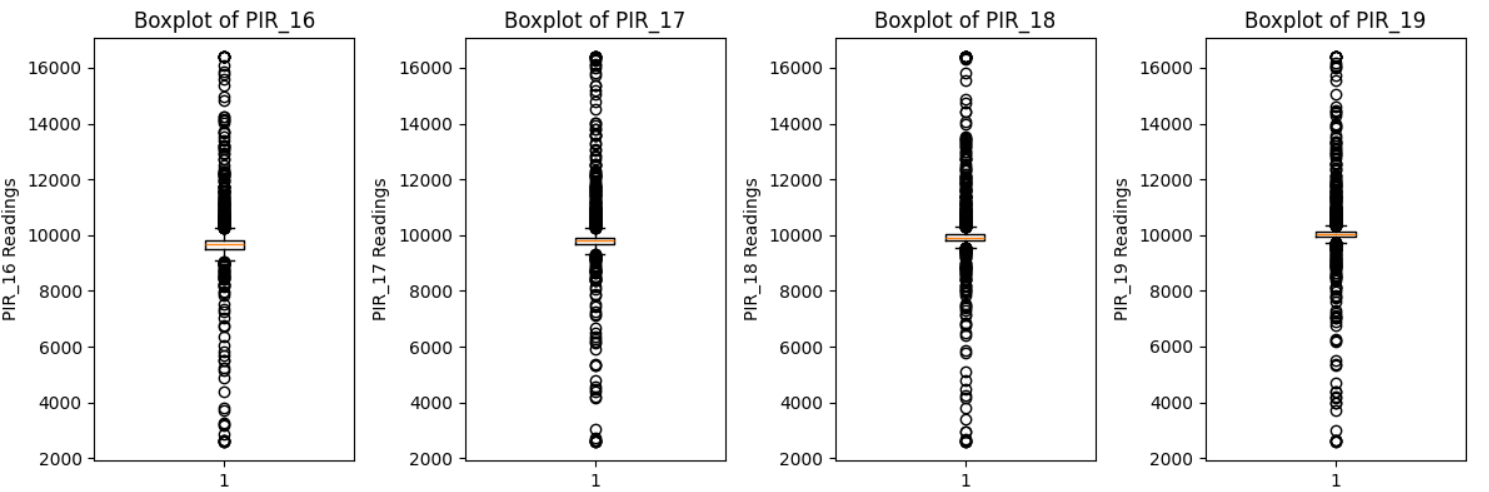
The next step is to visualize the data to further understand the distribution of individual variables and relationships between them. This would typically involve creating various plots such as scatter plots, box plots (after cleaning), and other relevant visualizations based on the data's nature.

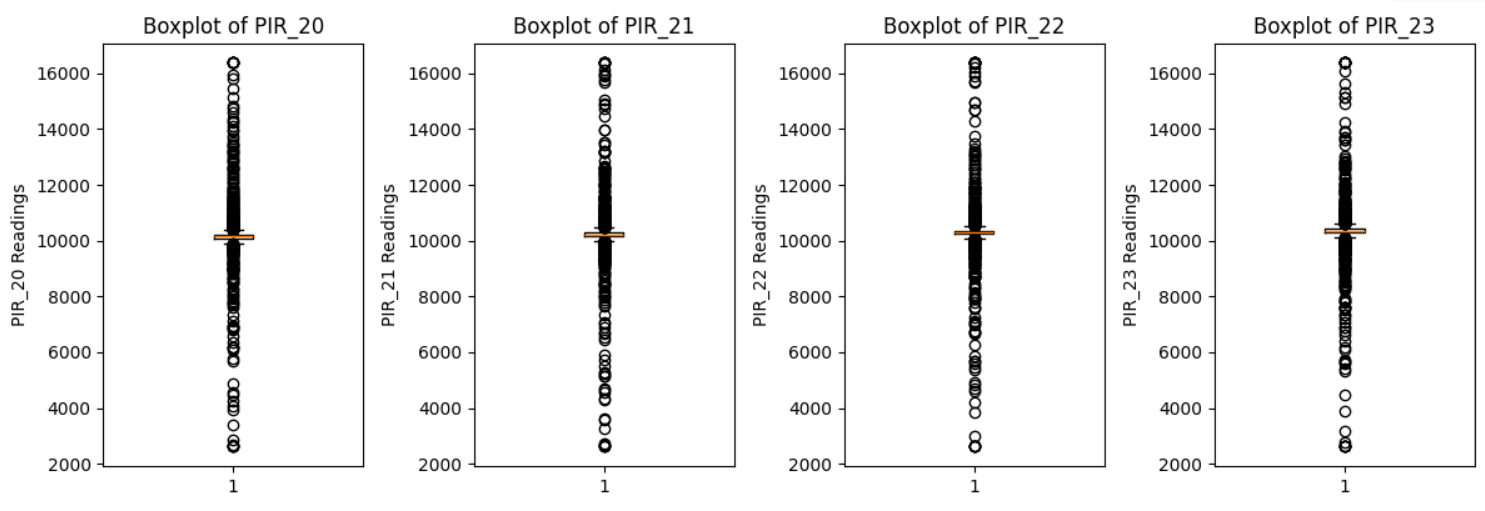


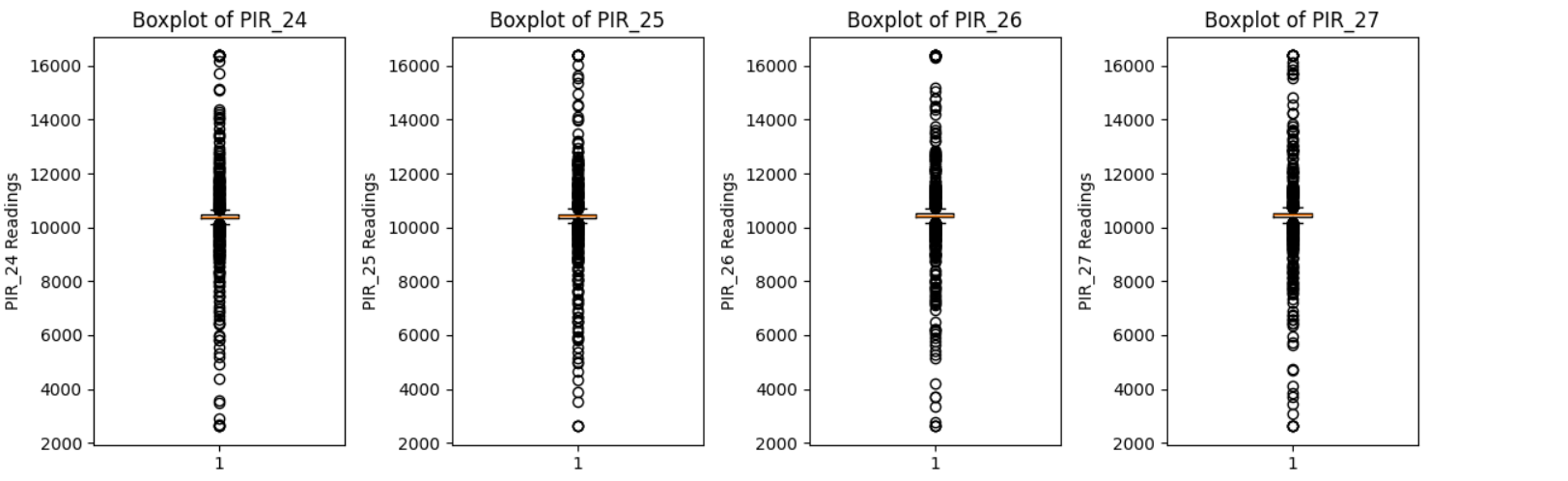


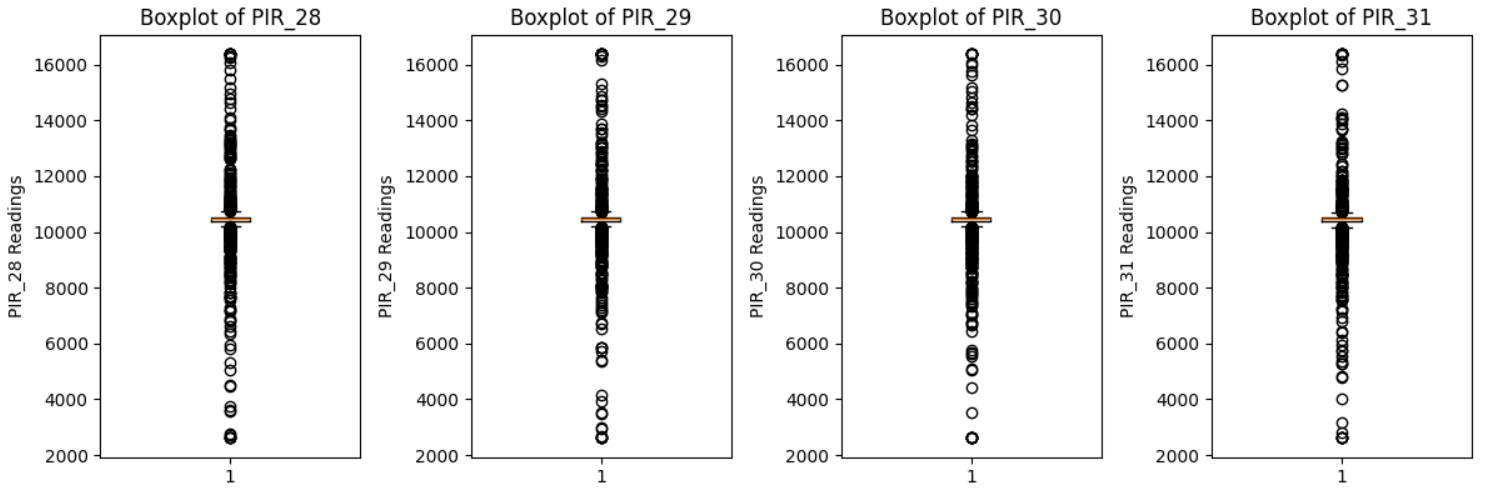


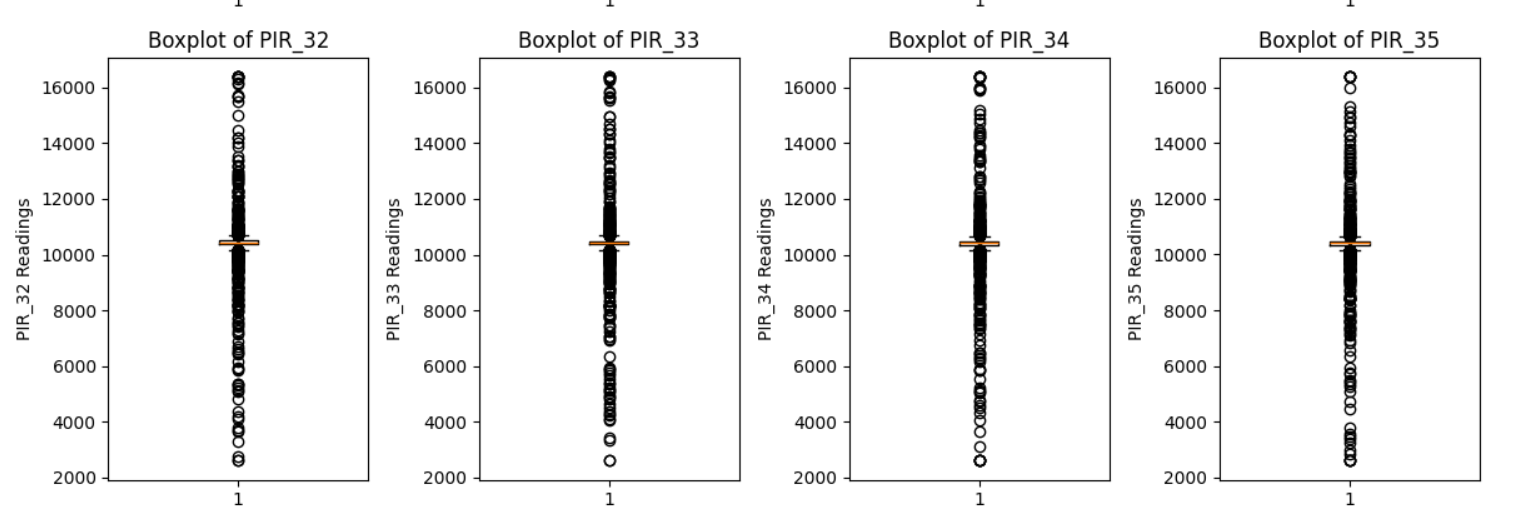


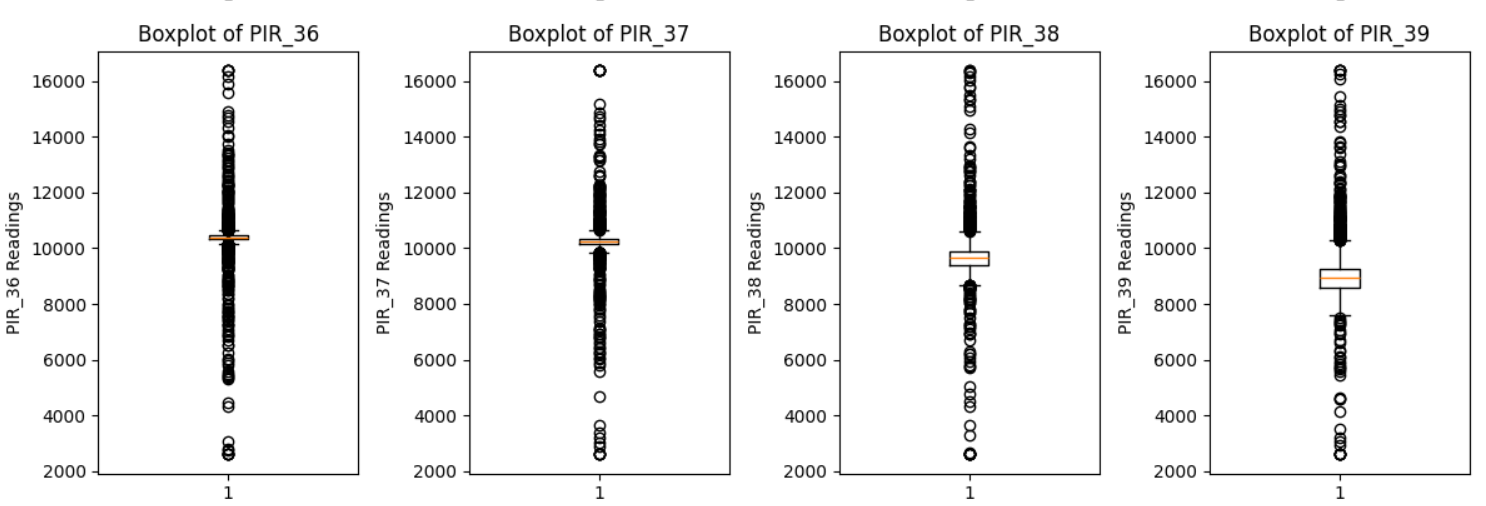


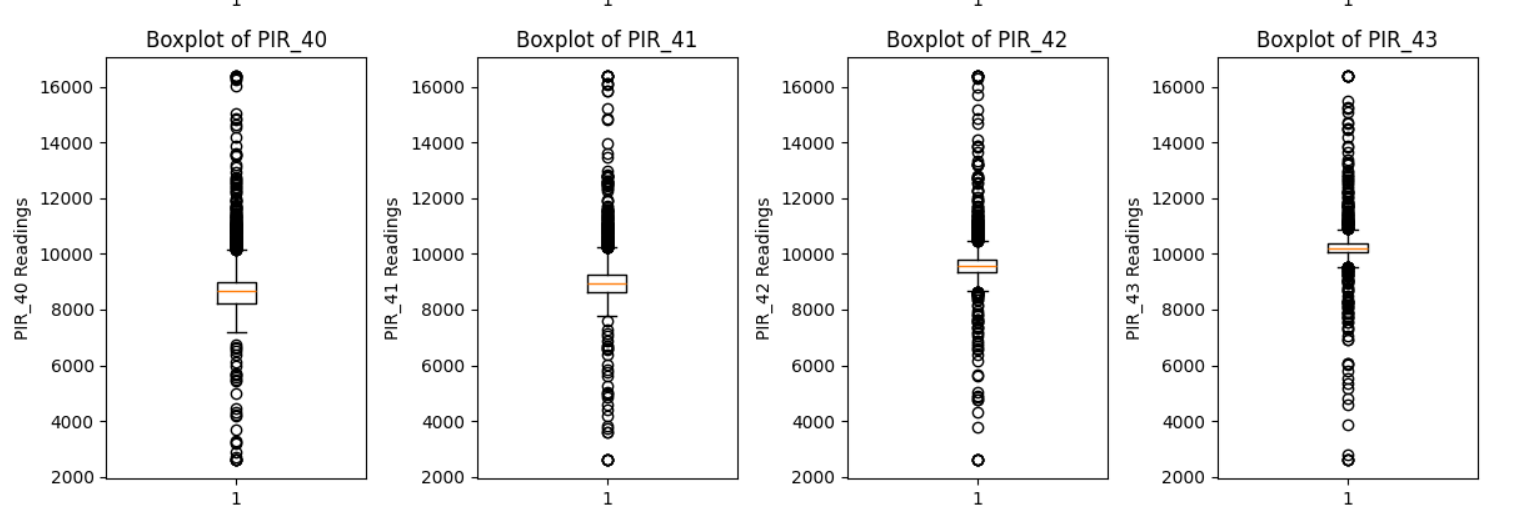


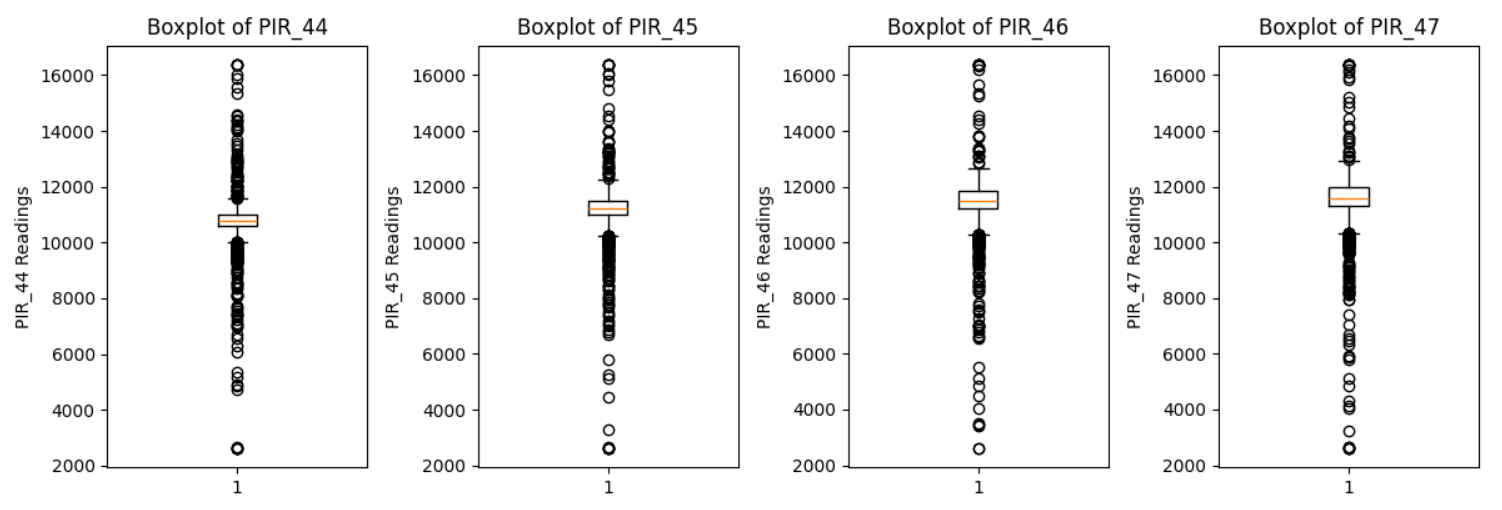


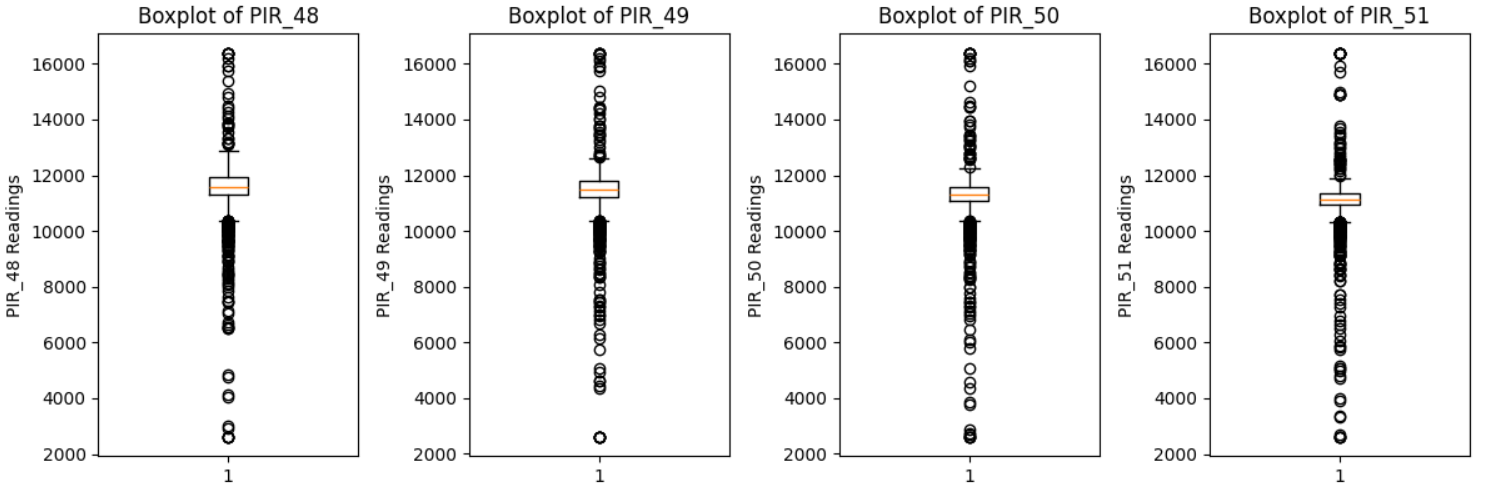


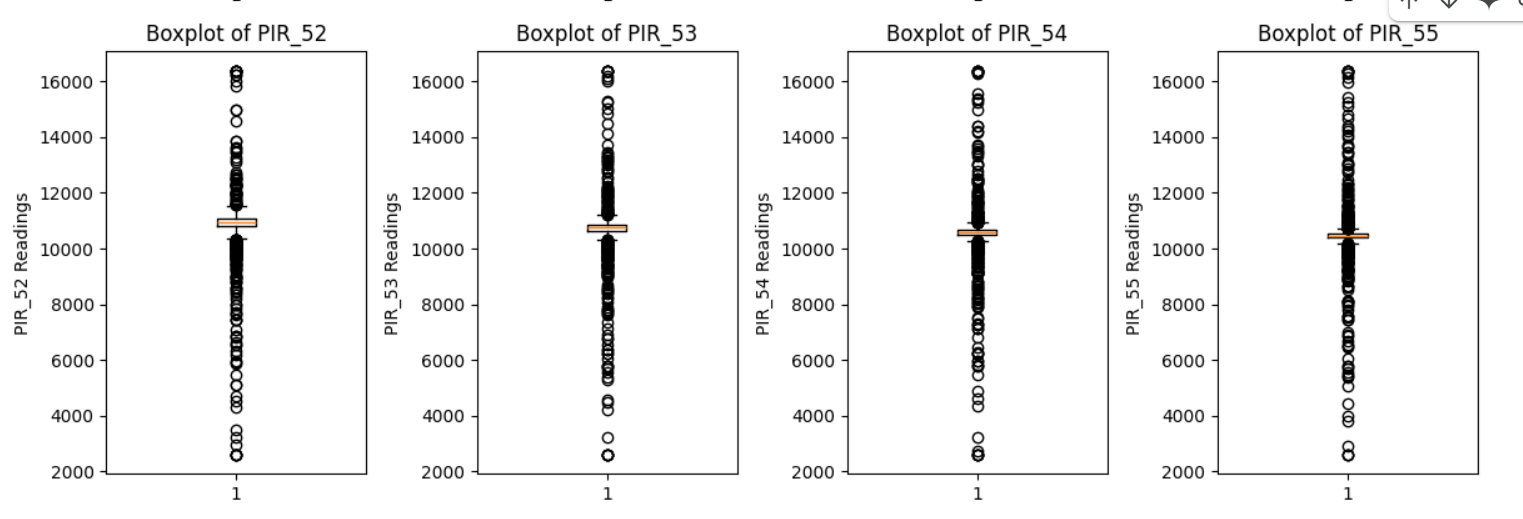












After generating the box plots for 'Temperature\_F' and all 'PIR' columns, we can evaluate them to gain insights into the distribution and potential outliers in these features.

**General observations from box plots:**

#### **1. Temperature\_F**

* The box plot shows a **very narrow interquartile range (IQR)**, indicating that indoor temperatures remained relatively constant.
* There are a few **outliers on the lower end**, possibly due to faulty readings or moments of ventilation.
* This limited variance suggests **low predictive value** unless correlated with specific events or time periods.

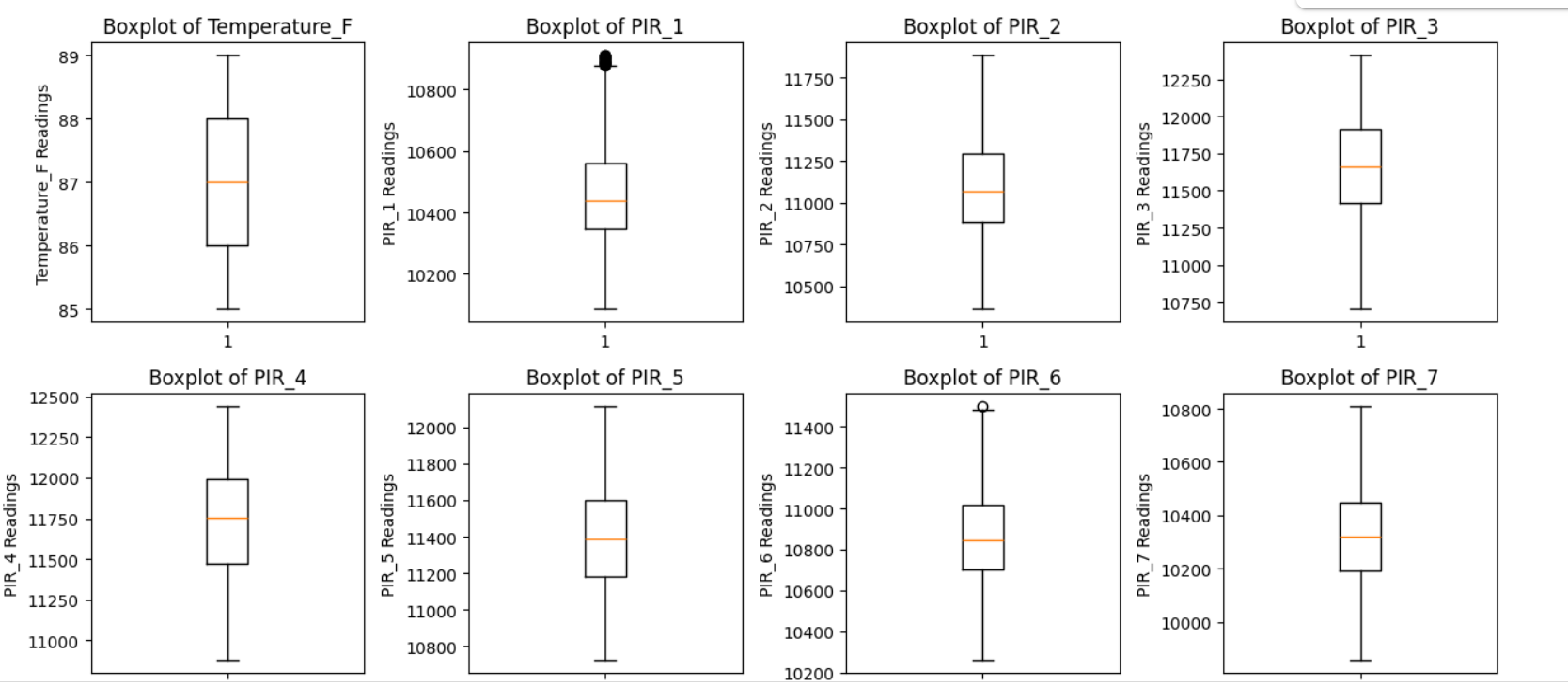
#### **2. PIR Sensor Readings (PIR\_1 to PIR\_55)**

* Most PIR sensors show **tight box plots with symmetric distributions**, centered between **8000–12000**.
* These patterns suggest **consistent activity levels** or baseline sensor outputs for most areas.
* Almost all sensors show **multiple outliers**, especially on the **lower end**, indicating moments of **no activity or significant deviations** (e.g., absence of occupants).
* A few PIR sensors like **PIR\_1**, **PIR\_6**, and **PIR\_13** display **extreme outliers**, which may point to:  
  + Malfunctioning sensors,
  + Unusual occupancy patterns,
  + Environmental disturbances.
* Some sensors (e.g., **PIR\_29**, **PIR\_31**, **PIR\_38**) show slightly **higher spread**, indicating more variability in activity at those locations.

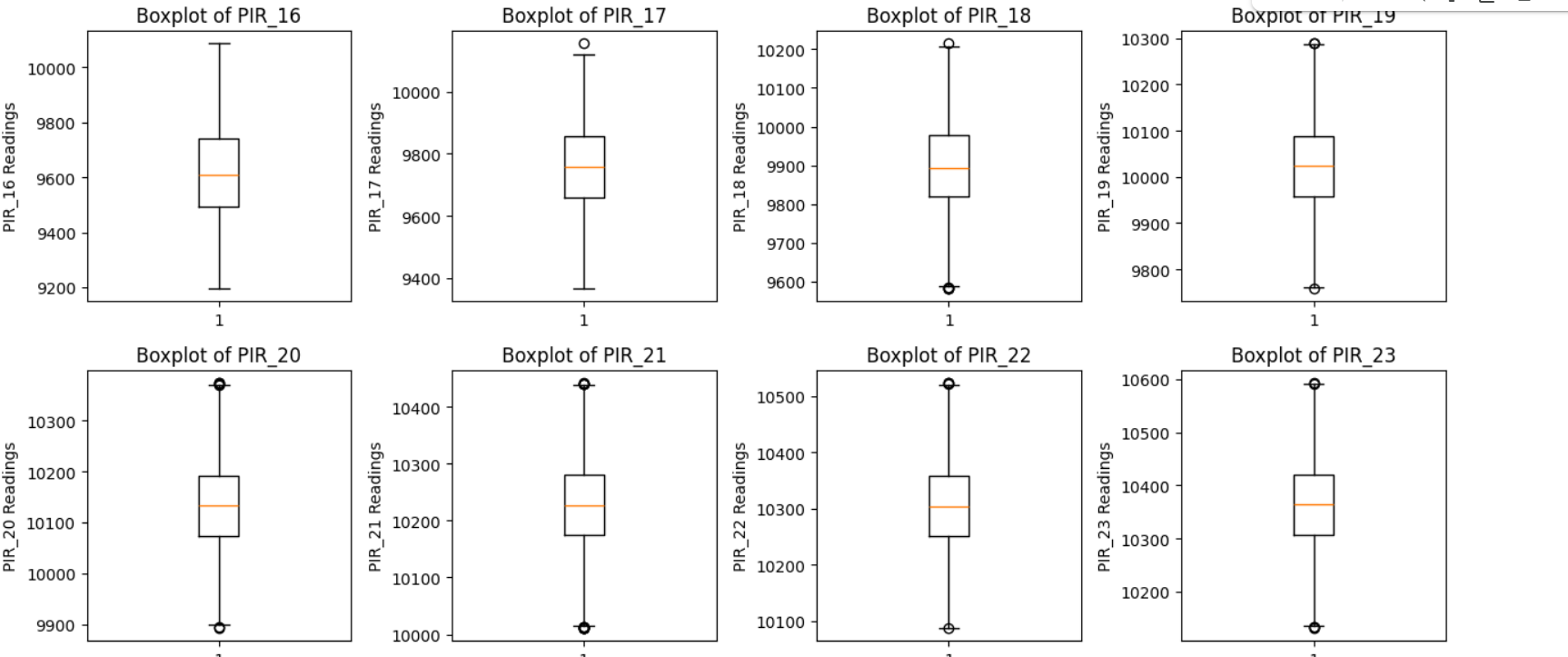
**Removing Outliers :**

We can see on the above visualization of our data set through box plot visualization we have so many outliers in out data set so to clean out data we have to remove theres outliers so that our data lie in one range

After implementing the outlier removal algorithm/code our data will look like this

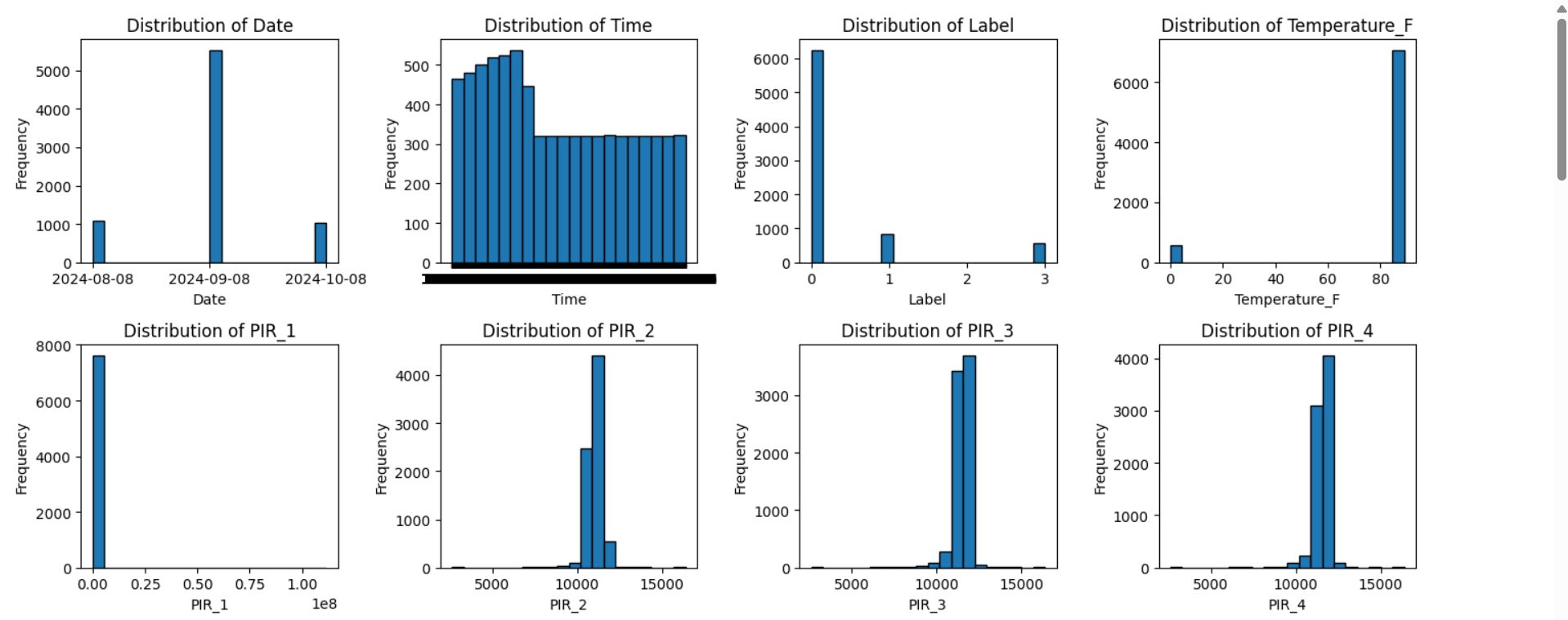


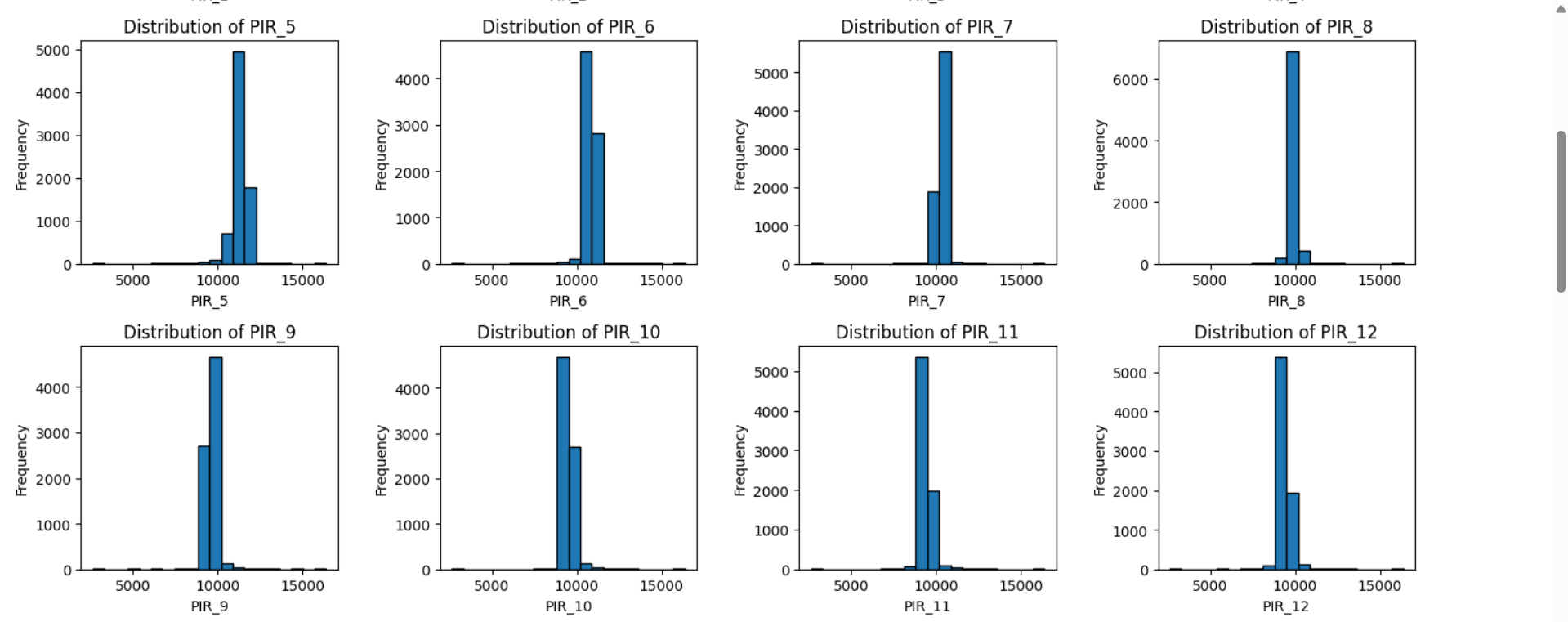


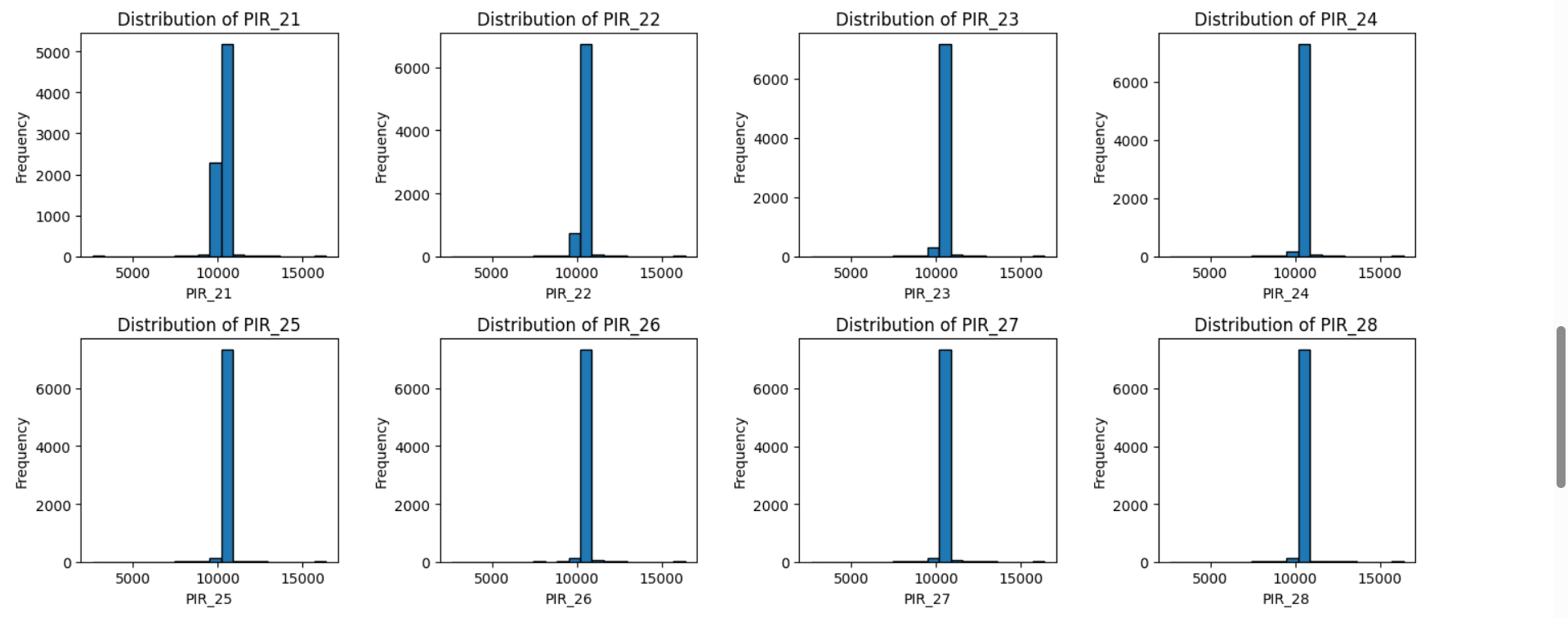


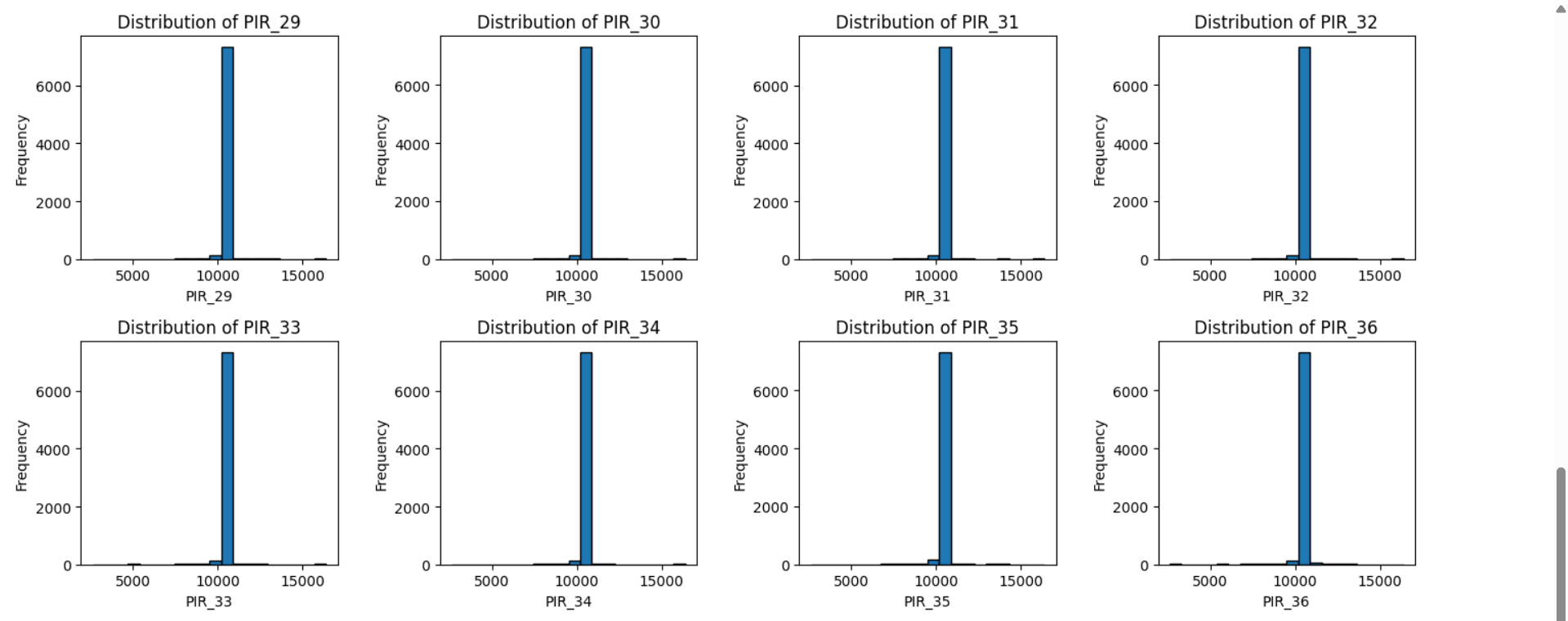
Further remaining box plots also shows that most of our outliers are removed from the data

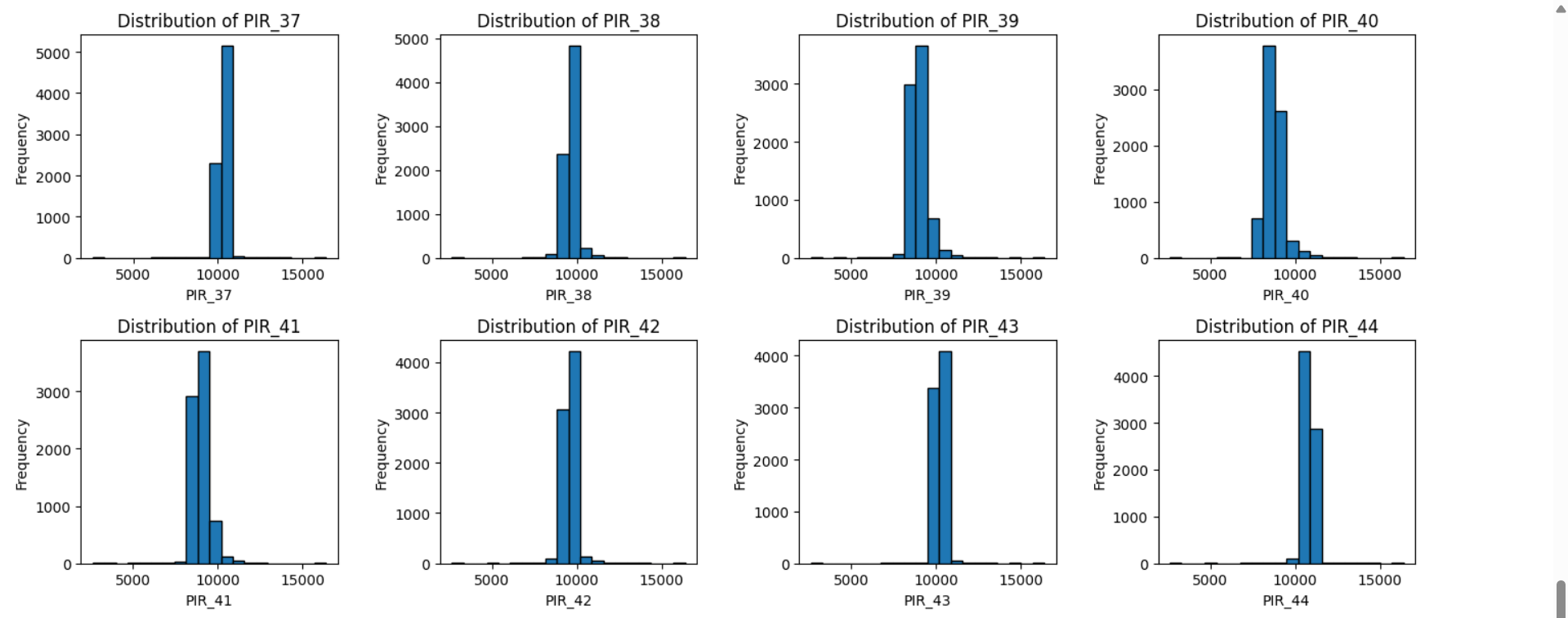
**Histogram:**

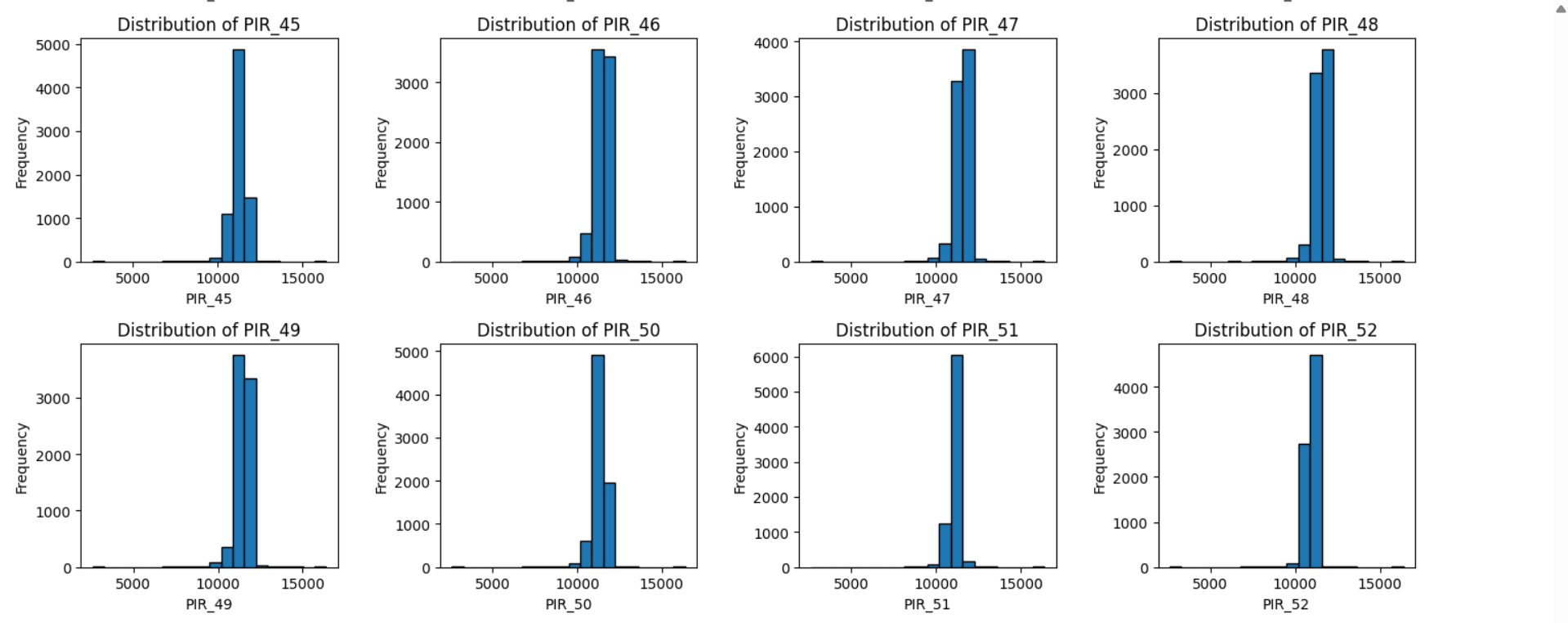


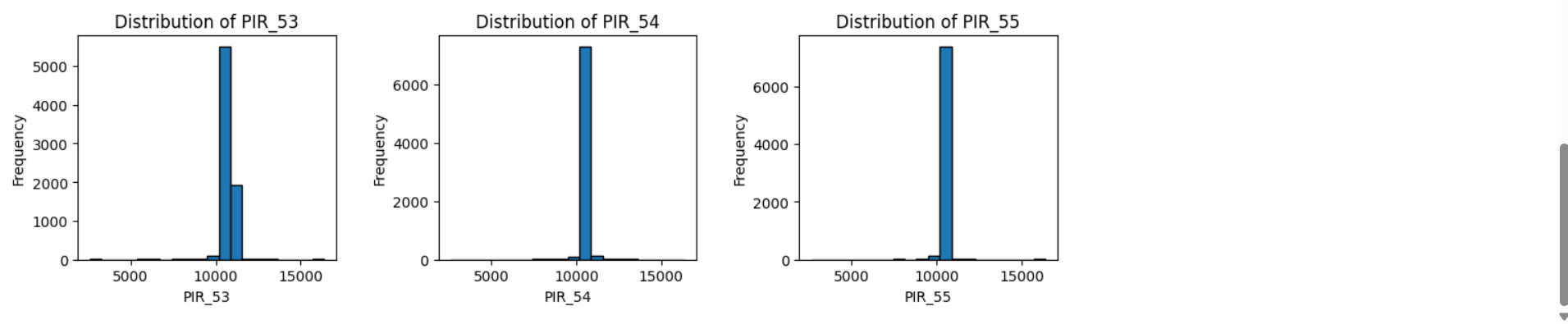












**Observations:**

* The data collection period is limited to a few days, and timestamps are evenly distributed across the day, indicating consistent hourly sampling.
* The label distribution is significantly imbalanced, with one class (likely '0') dominating. This imbalance must be addressed during modeling to avoid bias.
* Temperature readings remain within a narrow band (~68–78°F), reflective of controlled indoor conditions.
* PIR sensor data (PIR\_1 to PIR\_55) generally follow unimodal distributions, with some showing mild skewness. Most sensors record values around a central mean between 8000 and 12000, suggesting stable behavior. However, some sensors show outliers or very narrow distributions, which could either be indicative of sensor noise, placement, or low/no occupancy events.
* These observations underscore the importance of feature scaling, outlier treatment, and feature selection during preprocessing and modeling.

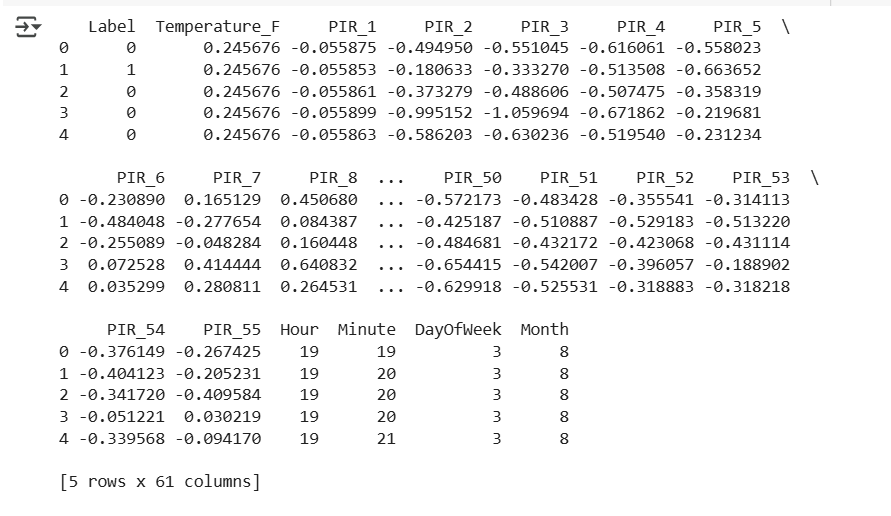
## **Data Transformation:**

#### **Date time feature extraction:**

The original dataset included Date and Time columns. These fields in raw format are not directly useful for machine learning models. We needed to extract meaningful time-based patterns that could influence human presence (e.g., time of day or day of week).

**What was done:**

* Extracted the following features:  
  + Hour – to capture daily activity trends
  + Minute – for finer time resolution
  + DayOfWeek – to differentiate weekdays vs. weekends
  + Month – to detect seasonal trends
* Removed the original Date and Time columns after feature extraction.



#### **Scaling numeric features:**

Sensor readings (PIR\_1 to PIR\_55) and Temperature\_F have varying ranges. Unscaled features can cause models (especially distance-based or gradient-based ones) to be biased toward higher-magnitude values.

**What was done:**

* Applied **StandardScaler** to:  
  + All 55 PIR sensor columns
  + Temperature\_F
* The target column Label (binary: 0/1) was kept as-is, since it does not need scaling.

#### **Feature selection:**

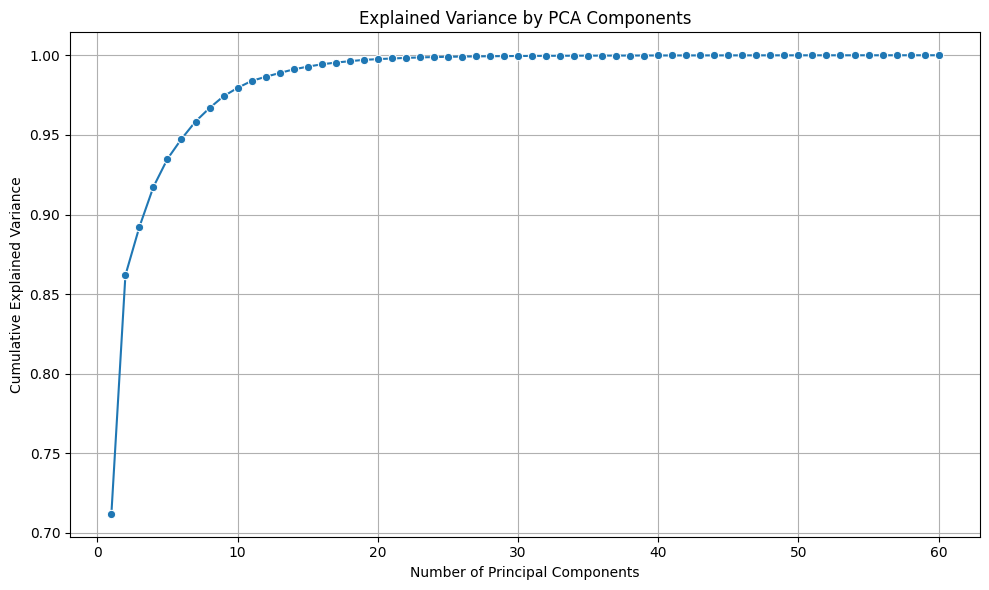
* There were **55 PIR features**, which increases computational complexity and risk of overfitting.
* Many of them may be **correlated or redundant**.
* We needed to reduce dimensions while preserving maximum information (variance).

**What was done:**

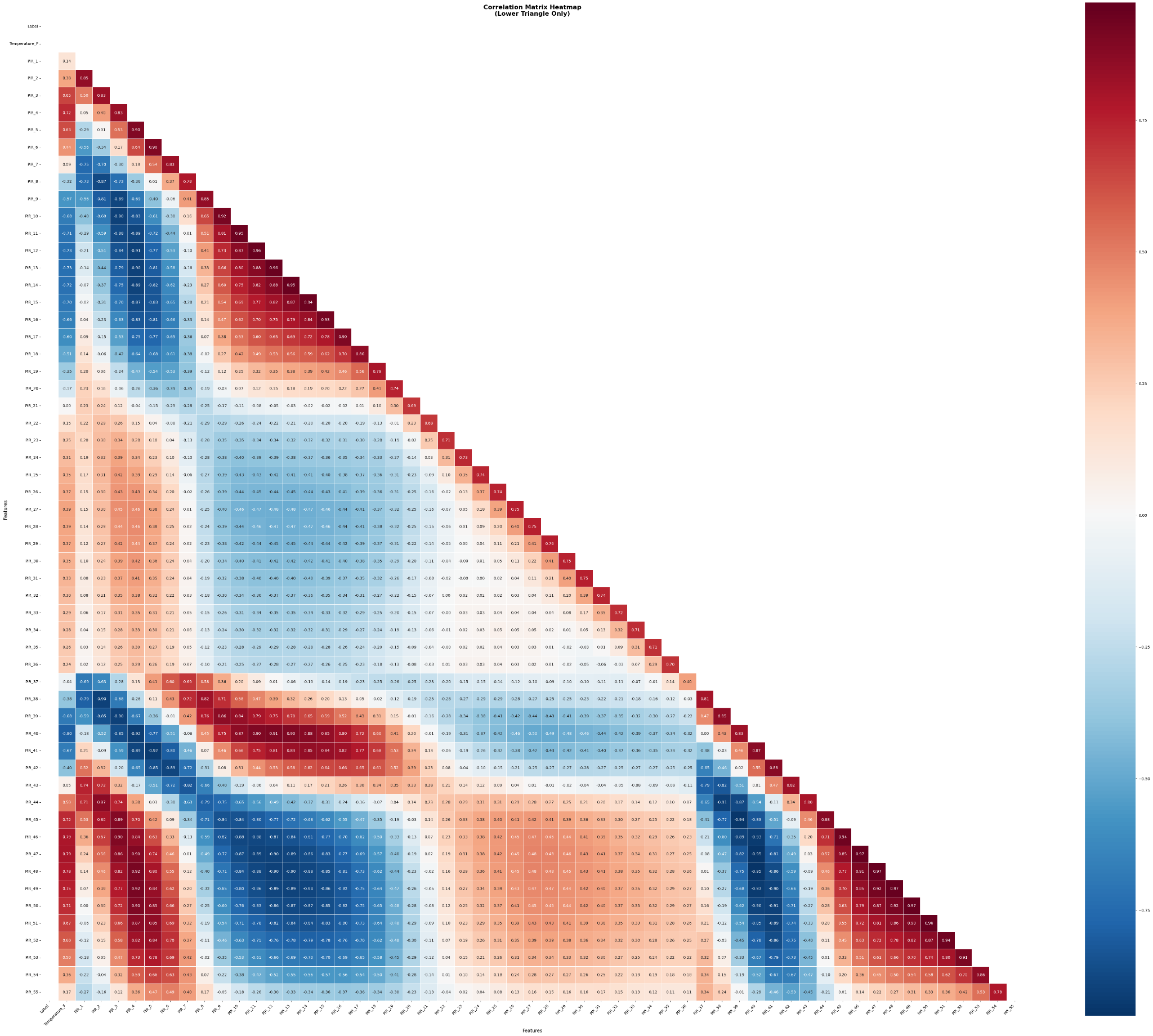
* Applied **Principal Component Analysis (PCA)** on the scaled dataset.
* Used n\_components=0.95 to automatically retain enough components to explain **95% of the variance**.
* Based on the explained variance plot, **around 10 components** were sufficient.

**Outcome:**

* Reduced the dataset’s dimensionality from ~60 features to ~10.
* Retained over **95% of the original variance**.
* Significantly improved **model efficiency** without major loss in information.



**Heatmap and correlation:**

****

The heatmap generated in provides a visual representation of the **Pearson correlation coefficients** between the numerical features in your dataset. Each cell in the heatmap shows the correlation value between two features. The color of the cell indicates the strength and direction of the correlation:

* **Colors near red:** Indicate a **strong negative correlation**. As one feature increases, the other tends to decrease.
* **Colors near blue:** Indicate a **strong positive correlation**. As one feature increases, the other also tends to increase.
* **Colors near white/light gray:** Indicate a **weak or no correlation**. The features do not have a clear linear relationship.
* **Patterns:** Look for blocks of similar colors. For example, a group of PIR sensors might show strong positive correlation with each other, forming a block of blue in the heatmap.

The **absolute value** of the correlation coefficient ranges from 0 to 1:

* **0:** No linear correlation.
* **1:** Perfect positive linear correlation.
* **-1:** Perfect negative linear correlation.

Analyzing the heatmap allows us to identify several key relationships within the data:

1. **Correlations between PIR Sensors:** Observe the squares where PIR sensor columns intersect. High positive correlations between different PIR sensors (e.g., PIR1 and PIR2) suggest that these sensors are detecting movement in similar areas or at similar times. This could indicate redundancy in these features or provide insights into the spatial arrangement of activity. Conversely, low correlations might mean the sensors are in different zones or capturing distinct events.
2. **Correlation between PIR Sensors and Temperature:** Look at the row/column where Temperature\_F intersects with the PIR sensor columns. A significant correlation here might suggest that temperature influences the PIR sensor readings or vice versa. However, direct correlation is less likely unless there's a very specific relationship not typically expected between motion detection and temperature. Any observed correlation might warrant further investigation to understand the underlying cause.
3. **Correlation between PIR Sensors and Label:** The Label column, which was likely encoded numerically in a previous step, represents a categorical outcome. The correlation between PIR sensors and the Label provides insight into how motion detected by each sensor relates to the different categories in the Label. A strong correlation (positive or negative depending on the encoding) indicates that the readings from that PIR sensor are highly associated with a particular label.
4. **Self-Correlation:** The diagonal of the correlation matrix (where a feature is correlated with itself) will always show a value of **1**. This is expected and confirms the matrix is correctly calculated.

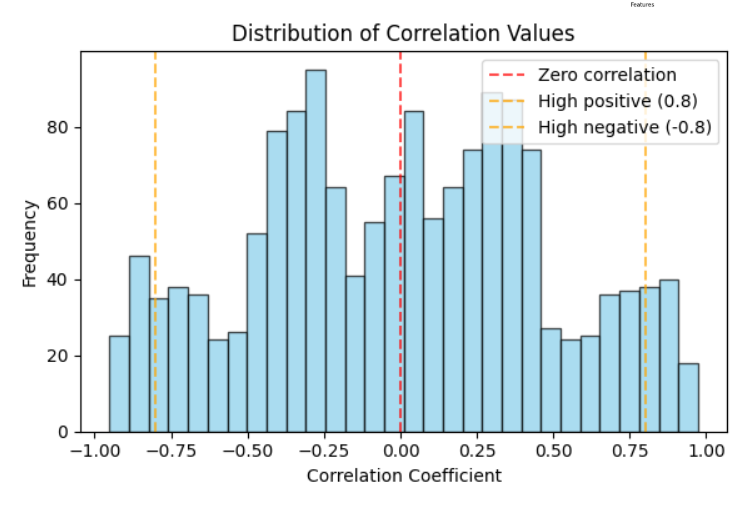
The code also explicitly identifies and prints **significant correlations** based on a defined threshold (defaulting to 0.5). This provides a more concrete list of feature pairs that have a correlation magnitude exceeding this threshold. You will see the output listing these pairs and their correlation values.

Additionally, the code specifically analyzes and prints the correlations between:

* PIR sensors and the Label column.
* PIR sensors and the Temperature\_F column.

These targeted results will highlight which PIR sensors, if any, have a strong linear relationship with the temperature and the target variable (Label).

**Correlation Distribution:**

****

## **Regression Models**

**Overview:**

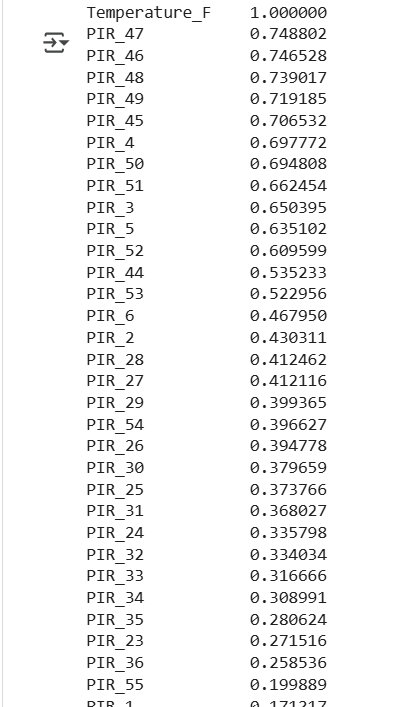
To evaluate the performance of various regression models, we adopted a systematic approach involving both cleaned and scaled versions of our dataset. Initially, the dataset was preprocessed to handle missing values, remove outliers, and ensure consistent data formatting—this version is referred to as the **cleaned data**. We then trained and evaluated multiple regression models, including Linear Regression, Polynomial Regression, and Decision Tree Regression, on this cleaned data.

In the next phase, we applied **feature scaling** techniques such as StandardScaler or MinMaxScaler to the cleaned dataset to normalize the input features—resulting in the **scaled data**. The same set of regression models were trained again using this scaled data.

We present and analyze the results in two parts: first, we discuss the models trained on the cleaned data, followed by a detailed look at those trained on the scaled data. Finally, we compare the performance metrics (such as R² score, MAE, MSE, and RMSE) of models from both stages to determine whether scaling had a significant impact on prediction accuracy.

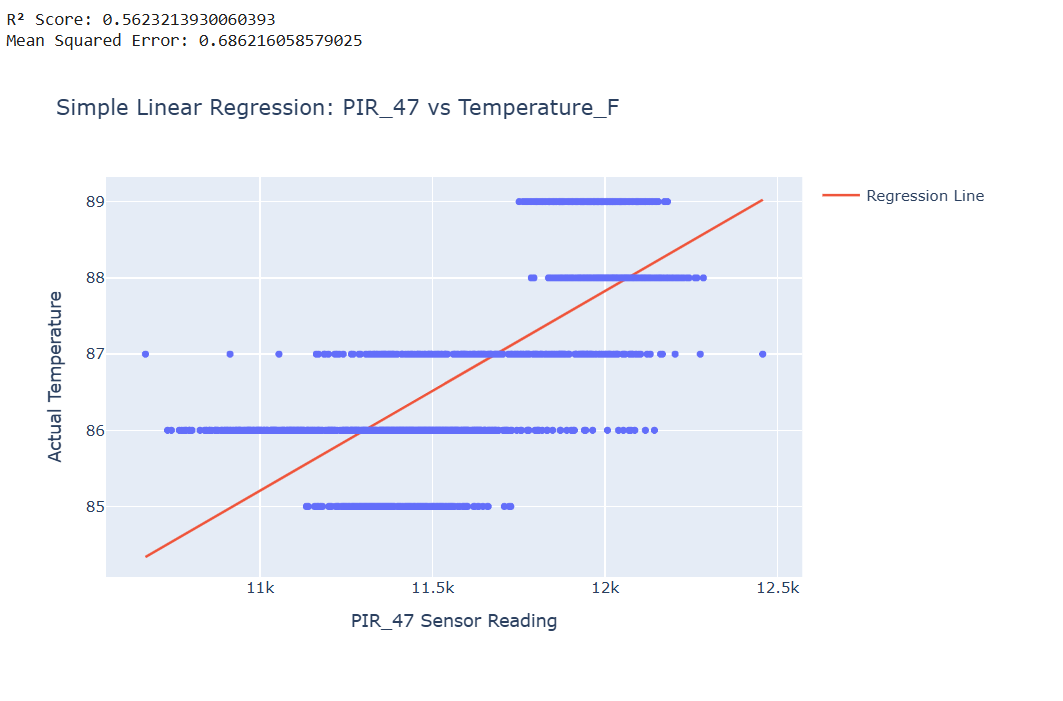
## **For Cleaned data:**

**To find highly correlated features with target variable ‘Temperature\_F’**

****

From this we can see that PIR\_47, PIR\_46, PIR\_48, PIR\_49, PIR\_45 are highly correlated with Temperature\_F, so we will use them to train our regression model.

### **Linear regression (Single Variable):**



#### **Model Performance:**

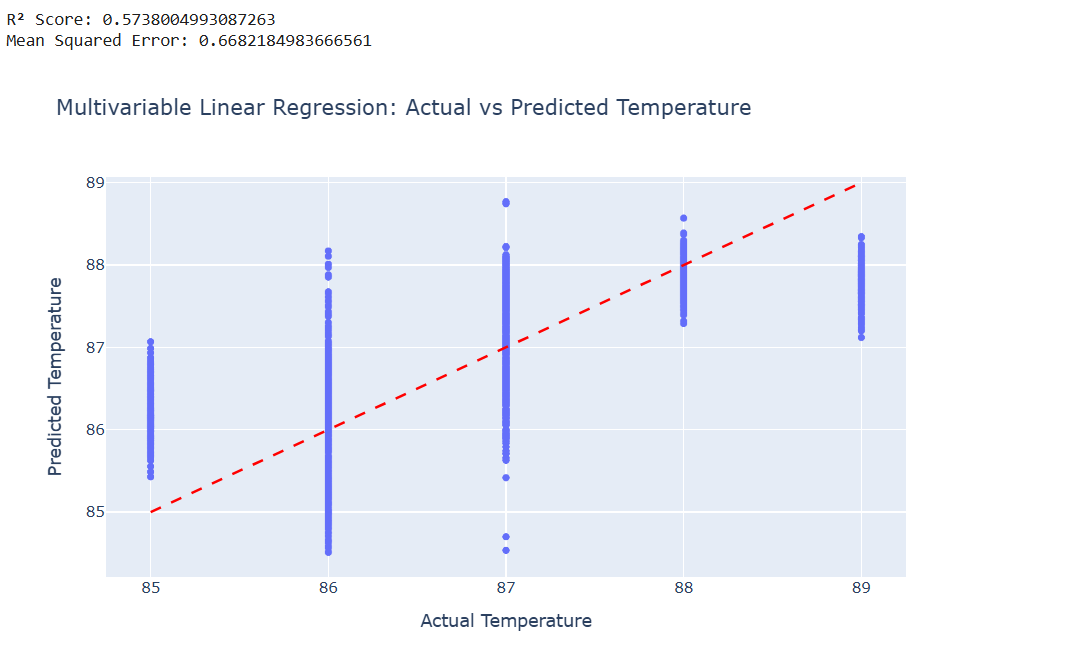
The R² score of 0.562 indicates that approximately 56% of the temperature variance is explained by the PIR\_47 sensor readings. This represents a moderate positive correlation - there's a clear relationship, but other factors also influence temperature.

The Mean Squared Error of 0.686 suggests the model's predictions deviate from actual values by roughly 0.83°F on average (square root of MSE), which is relatively small given your temperature range of about 85-89°F.

PIR\_47 demonstrates a moderately strong positive correlation with temperature, making it a useful but not perfect predictor. The relationship is statistically significant and practically meaningful for temperature estimation applications. However, the discrete temperature clustering suggests you may want to investigate whether higher resolution temperature measurements could improve model accuracy.

The sensor appears to respond predictably to temperature changes, supporting its reliability for thermal monitoring applications within the tested range.

### **Linear regression (Multivariable):**



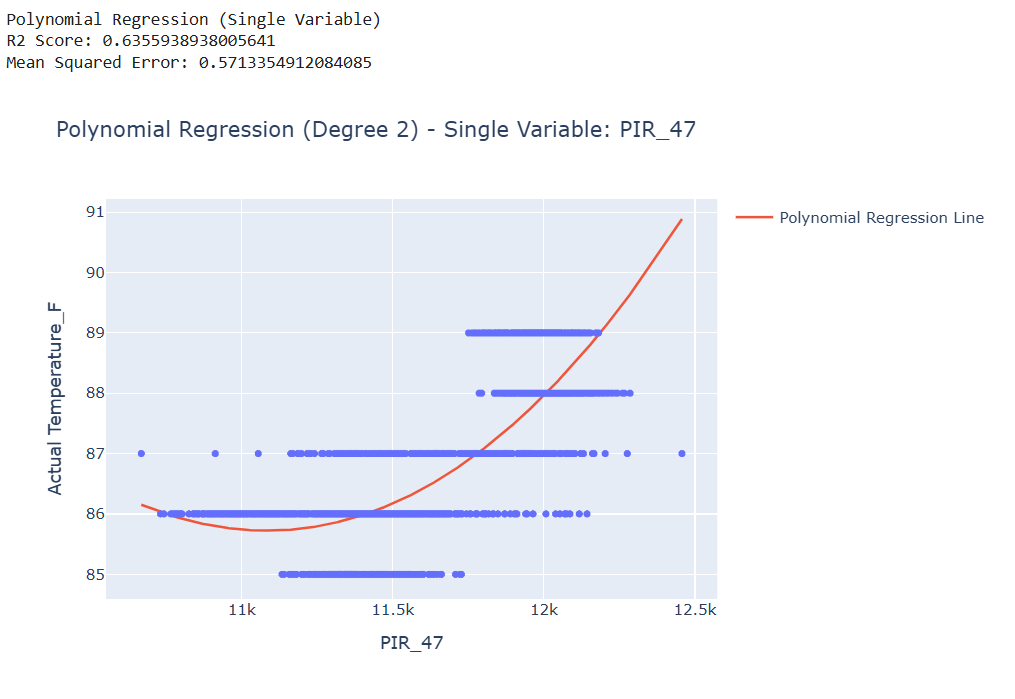
#### **Model performance:**

The multivariable model shows a slight improvement over the single-variable PIR\_47 model:

* R² increased from 0.562 to 0.574, meaning the model now explains 57.4% of temperature variance
* MSE decreased slightly from 0.686 to 0.668, indicating marginally better prediction accuracy
* The improvement is modest (about 1.2% increase in explained variance), suggesting the additional variables provide some but limited predictive value

The multivariable approach provides incremental improvement over using PIR\_47 alone, but the gains are marginal. The model's predictive capability remains moderate, with the discrete nature of your actual temperature measurements likely limiting overall performance. The wide prediction intervals suggest the model may benefit from additional features or a different modeling approach.

### **Polynomial Regression (Single Variable):**



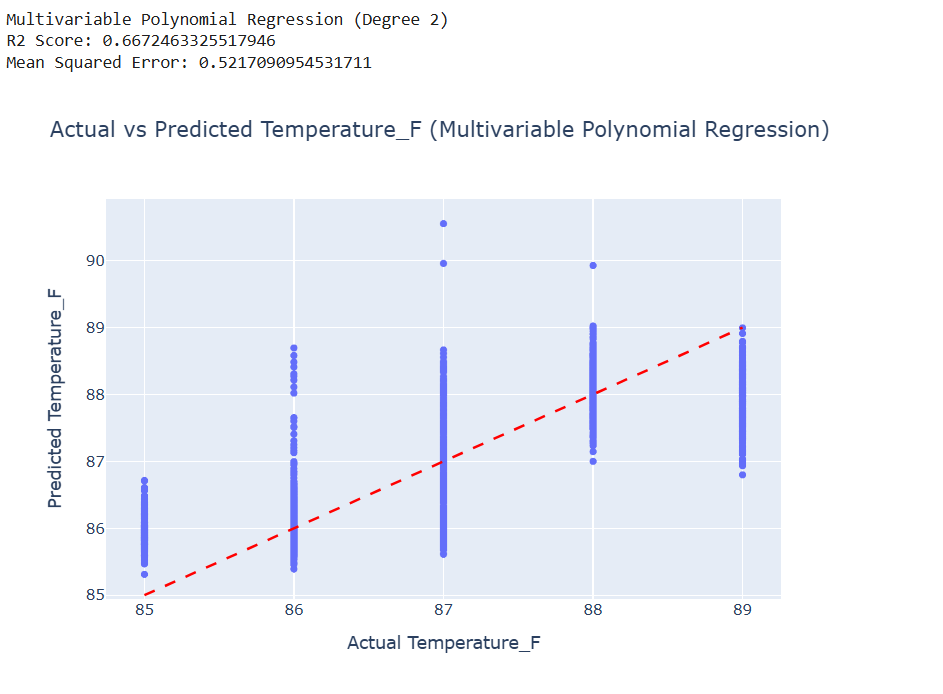
#### **Model Performance:**

The polynomial regression shows substantial improvement over the linear models:

* **R² increased to 0.636**, explaining 63.6% of temperature variance (compared to 56.2% for simple linear and 57.4% for multivariable linear)
* **MSE decreased to 0.571**, representing the best prediction accuracy among all models tested
* This represents a **13% improvement** in explained variance over the simple linear model

The polynomial regression demonstrates that the relationship between PIR\_47 and temperature contains significant non-linear components that linear models cannot capture. This finding suggests that PIR sensors may exhibit more complex thermal response characteristics than initially assumed, and non-linear modeling approaches should be considered for optimal temperature prediction accuracy in sensor-based systems.

### **Polynomial Regression (Multivariable):**



#### **Model Performance:**

The multivariable polynomial regression delivers the strongest performance across all models tested:

* **R² of 0.667** - explaining 66.7% of temperature variance, the highest among all approaches
* **MSE of 0.522** - the lowest prediction error achieved
* Represents a **5% improvement** over single-variable polynomial and **19% improvement** over simple linear regression

The multivariable polynomial approach successfully combines the benefits of multiple sensor inputs with non-linear modeling capabilities. This represents the optimal balance of model complexity and predictive power for the sensor data. The results suggest that temperature relationships in the sensor system involve both multiple variables and non-linear interactions, making this the most appropriate modeling approach for practical temperature prediction applications.

## 

### **Decision Tree regressor:**





#### **Model Performance:**

The decision tree model shows notably weaker performance compared to your other approaches:

* **R² of 0.42** - explaining only 42% of temperature variance, the lowest among all models tested
* **MSE of 0.91** - the highest prediction error, indicating less accurate predictions
* Represents a **significant decline** from your best model (multivariable polynomial with R² = 0.667)

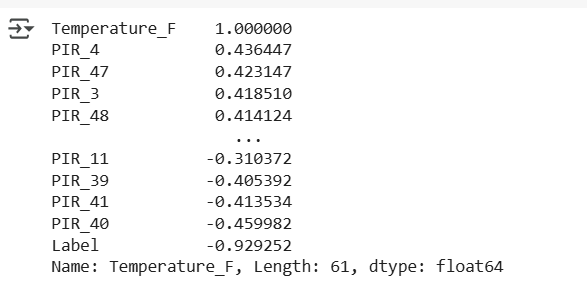
The decision tree's discrete nature creates a "stepped" prediction pattern, which doesn't capture the continuous temperature relationships that your sensor data exhibits. This suggests the tree-based approach may be overly simplistic for your dataset's complexity.

## **Comparative Model Performance Ranking**

1. **Multivariable Polynomial (R² = 0.667, MSE = 0.522)** - Best
2. Single Variable Polynomial (R² = 0.636, MSE = 0.571)
3. Multivariable Linear (R² = 0.574, MSE = 0.668)
4. Simple Linear (R² = 0.562, MSE = 0.686)
5. **Decision Tree (R² = 0.42, MSE = 0.91)** - Worst

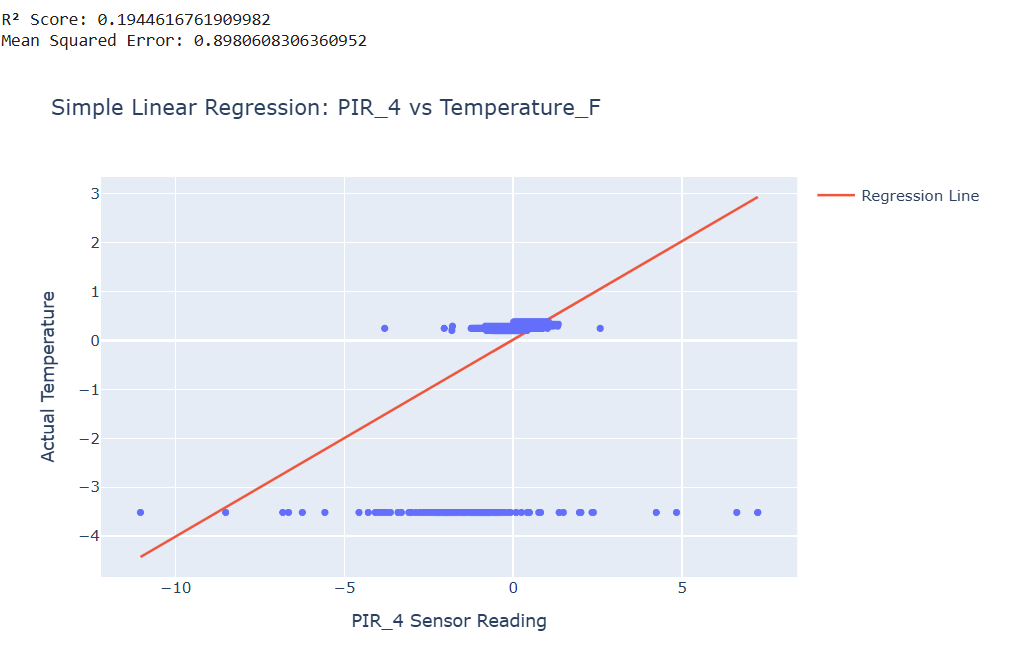
## **For Scaled data:**

**To find highly correlated features with target variable ‘Temperature\_F’**

****

From this we can see that PIR\_47, PIR\_4, PIR\_48, PIR\_3 are highly correlated with Temperature\_F, so we will use them to train our regression model.

### **Linear regression (Single Variable):**



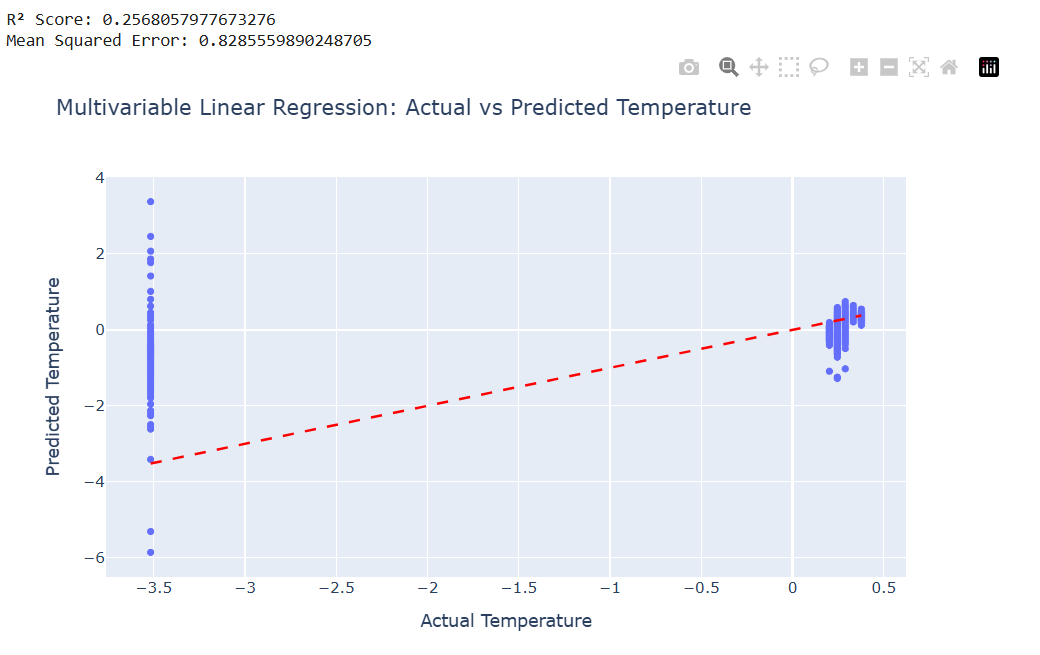
#### **Model Performance:**

The simple linear regression on scaled data shows poor performance:

* **R² of 0.194** - explaining only 19.4% of temperature variance
* **MSE of 0.898** - indicating high prediction error
* This represents **significantly weaker performance** compared to PIR\_47 models

The scaling transformation reveals that **PIR\_4 is a poor predictor of temperature** compared to PIR\_47. This finding suggests that not all PIR sensors in the system have equal predictive value for temperature estimation. The weak correlation indicates PIR\_4 may be measuring different environmental conditions or have different calibration characteristics.

### **Linear regression (Multivariable):**



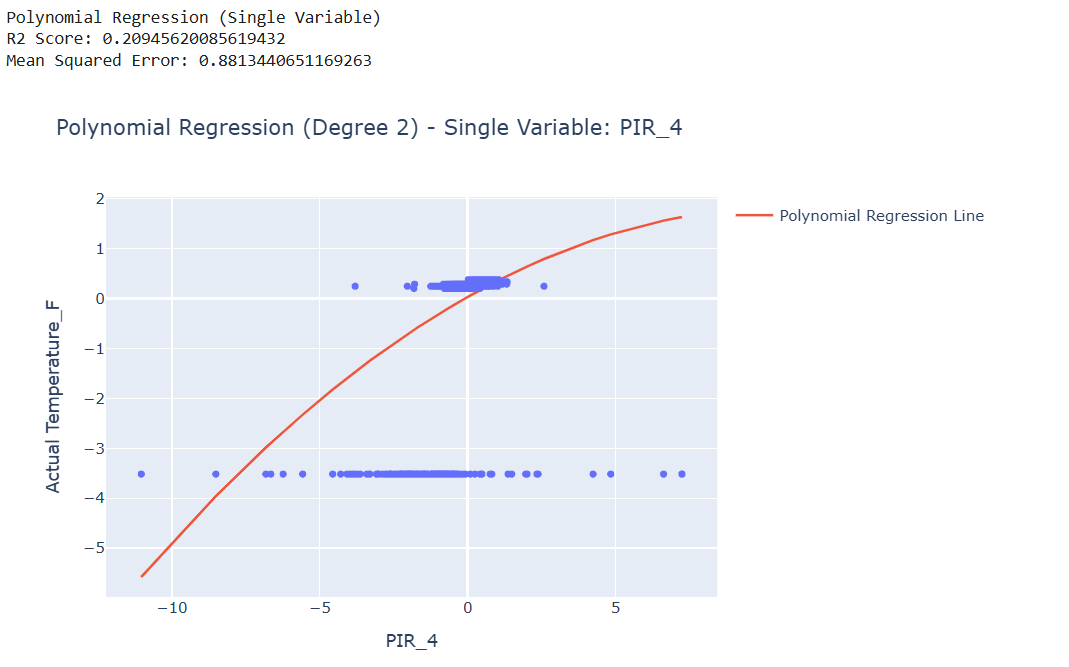
#### **Model performance:**

The multivariable approach on scaled data shows improved but still limited performance:

* **R² of 0.257** - explaining 25.7% of temperature variance
* **MSE of 0.826** - moderate prediction error
* **Improvement over single PIR\_4**: 6.3% increase in explained variance (from 19.4% to 25.7%)

The scaled multivariable model demonstrates that while additional variables provide some predictive improvement, the overall sensor array still shows **weak correlation with temperature** when PIR\_47 is not the primary predictor. The standardized scaling reveals that the prediction challenges are not simply due to scale differences between variables.

### **Polynomial Regression (Single Variable):**



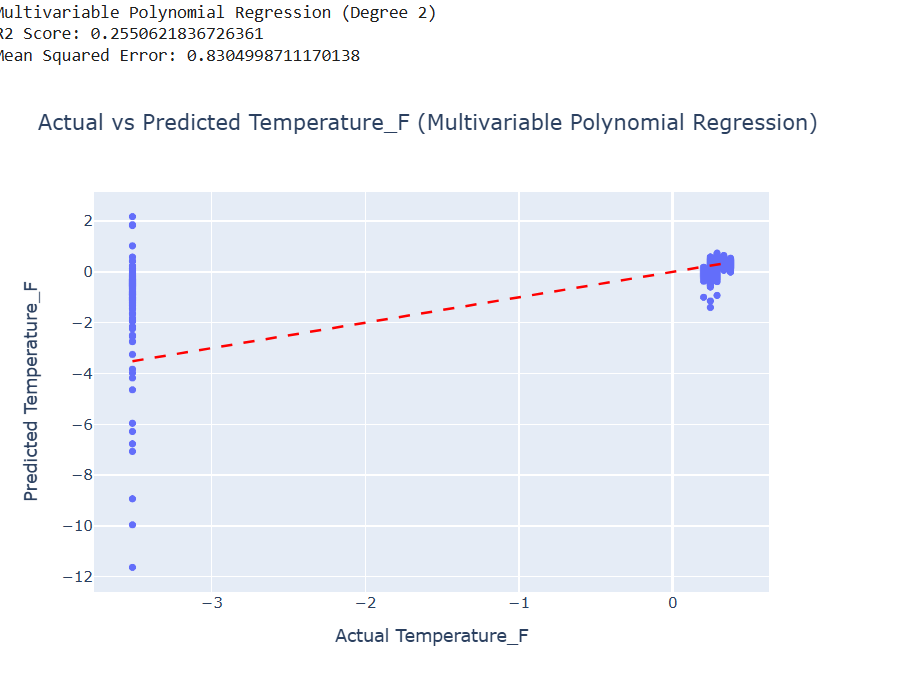
#### **Model Performance:**

The polynomial approach shows minimal improvement over linear modeling:

* **R² of 0.209** - explaining only 20.9% of temperature variance
* **MSE of 0.881** - similar prediction error to linear models
* **Marginal improvement**: Only 1.5% increase over simple linear PIR\_4 (19.4% to 20.9%)

The polynomial curve passes through empty regions where no data exists.Data clustering suggests the relationship may be more categorical than continuous.The curve doesn't improve prediction accuracy for the clustered data points

### **Polynomial Regression (Multivariable):**



#### **Model Performance:**

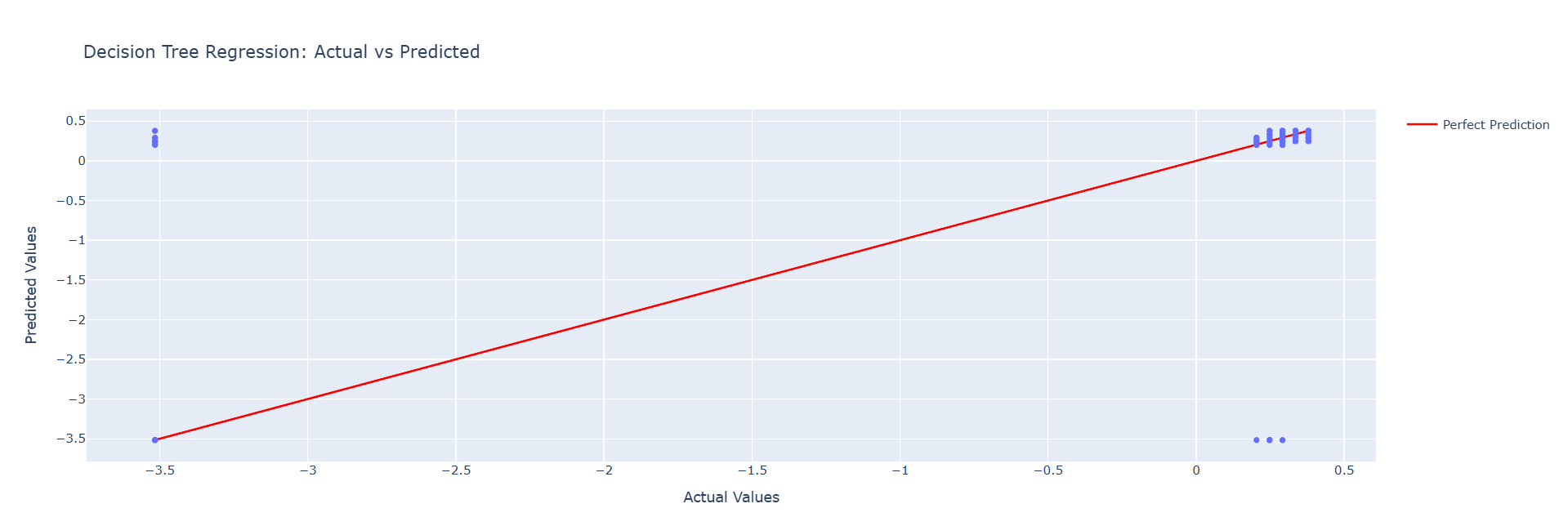
**R² Score: 0.255** - explaining 25.5% of temperature variance

**MSE: 0.830** - indicating moderate prediction error

This represents the best performance among the PIR\_4-based models.The multivariable polynomial approach, while showing the best R² score for PIR\_4, produces unreliable predictions with extreme values that fall outside physically meaningful temperature ranges. This demonstrates that higher R² scores don't always indicate better practical model performance when prediction reliability is considered.

### **Decision Tree regressor:**





#### **Model Performance:**

The decision tree shows dramatically better performance on scaled data compared to previous PIR\_4 models:

* **R² Score: 0.53** - explaining 53% of temperature variance (more than double the polynomial model's 25.5%)
* **MSE: 0.52** - substantially lower prediction error compared to 0.830 for polynomial regression
* This represents the **best performing model** for PIR\_4 sensor data

The decision tree's success with PIR\_4 data suggests that the temperature monitoring system may have threshold-based or categorical operating states rather than continuous temperature variation. This finding has important implications for sensor calibration and system design.

## **Comparative Model Performance Ranking**

1. **Decision Tree: R² = 0.53, MSE = 0.52** - Best
2. Multivariable Polynomial: R² = 0.255, MSE = 0.830
3. Multivariable Linear: R² = 0.257, MSE = 0.826
4. Single Variable Models: R² < 0.21

## **Comparative Analysis: Cleaned vs Scaled Data Performance**

**Cleaned Data (PIR\_47) - High Performance:**

* Models achieve R² values from 0.42 to 0.667
* Multivariable Polynomial leads with 66.7% variance explained
* Continuous improvement from linear to polynomial approaches
* Decision trees perform poorly (worst at R² = 0.42)

**Scaled Data (PIR\_4) - Limited Performance:**

* Models achieve much lower R² values (maximum 0.53)
* Decision Tree surprisingly becomes the best performer
* Polynomial/Linear models show severe degradation
* Overall predictive capability remains weak

## **Key Insights from the Comparison**

**1. Sensor Quality Impact:**

* **PIR\_47 (cleaned data):** Strong temperature correlation enables effective modeling
* **PIR\_4 (scaled data):** Weak temperature correlation limits all modeling approaches
* Performance gap: 66.7% vs 53% maximum R² scores

**2. Data Preprocessing Effects:**

* **Scaling Impact:** Reveals fundamental data structure differences
* **Cleaned vs Scaled:** Different preprocessing approaches suit different sensor types
* **Model Selection:** Preprocessing choice should align with data characteristics

**Model Performance Hierarchy:**

* **Best Overall:** Multivariable Polynomial on PIR\_47 (R² = 0.667)
* **Best for Categorical Data:** Decision Tree on PIR\_4 (R² = 0.53)

## **Classification models on cleaned data:**

### **Data preparation:**

This section prepares your data for machine learning by separating the input features from the output target variable. The target variable, named 'Label', is extracted into its own variable target. All other columns in the original DataFrame df are considered features and are stored in the features DataFrame. This clear separation is essential for training supervised learning models, where you use the features to predict the target.

Following the separation, the code identifies any categorical columns within the features DataFrame, which are columns with a data type of 'object'. These categorical columns need to be converted into a numerical format for most machine learning algorithms. One-hot encoding is applied using pd.get\_dummies. This process transforms each categorical column into multiple binary columns, where each new column represents a unique category from the original column. The drop\_first=True argument is used during encoding to prevent multicollinearity. The resulting DataFrame, features\_encoded, contains the original numerical features plus the newly created binary features, making the data suitable for model training. The first few rows of features\_encoded are then displayed to show the result of this transformation.

### **Data Splitting:**

For models implementation we use to preparing the data for the machine learning models by splitting it into distinct sets for training and testing. This is a fundamental step in any supervised learning task. The goal is to train the models on a portion of the data and then evaluate their performance on a separate, unseen portion of the data. This helps to ensure that the models can generalize well to new data and are not simply memorizing the training examples (a phenomenon known as overfitting).

The code first imports the necessary function for this task, train\_test\_split, from the sklearn.model\_selection module. scikit-learn is a widely-used Python library for machine learning, providing a range of tools for tasks like classification, regression, clustering, and more. The model\_selection part specifically contains utilities for evaluating and selecting machine learning models.

The core of this section is the call to the train\_test\_split function. It takes the features\_encoded DataFrame (which contains the input variables after necessary preprocessing) and the target Series (the output variable we want to predict) as its main inputs. The test\_size=0.25 argument specifies that 25% of the entire dataset should be reserved for the testing set, while the remaining 75% will be used for training the models. The random\_state=42 argument is crucial for reproducibility. By setting a specific integer value for random\_state, the random splitting process will be the same every time the code is executed. This consistency is important for comparing the performance of different models or experiments. The function then returns four variables: X\_train and y\_train for the training features and target, and X\_test and y\_test for the testing features and target.

Finally, the code prints the shapes of the four resulting datasets using the .shape attribute. This is a quick way to confirm that the data has been split as expected and to see the number of samples and features in each set. For example, X\_train.shape will show the number of rows (training samples) and columns (features) in the training features DataFrame. Seeing these dimensions helps in understanding the scale of the training and testing data being used.

### **Model Training:**

Now for focusing on training several different classification models, It starts by importing the necessary model classes, such as LogisticRegression, DecisionTreeClassifier, RandomForestClassifier, SVC (for SVM), KNeighborsClassifier (for KNN), and GaussianNB (for Naive Bayes), all from the sklearn library. Then, it creates an instance for each of these models, initializing them with their default settings (except for LogisticRegression, where the maximum number of iterations is increased to 1000 to help with convergence). The core part of this section is the training process, which is performed by calling the fit() method on each model instance, using the training data (X\_train and y\_train). This step allows each model to learn the relationships between the features and the target variable from the provided data. Finally, a message is printed to confirm that all the models have been successfully trained, preparing them for subsequent evaluation and prediction tasks.

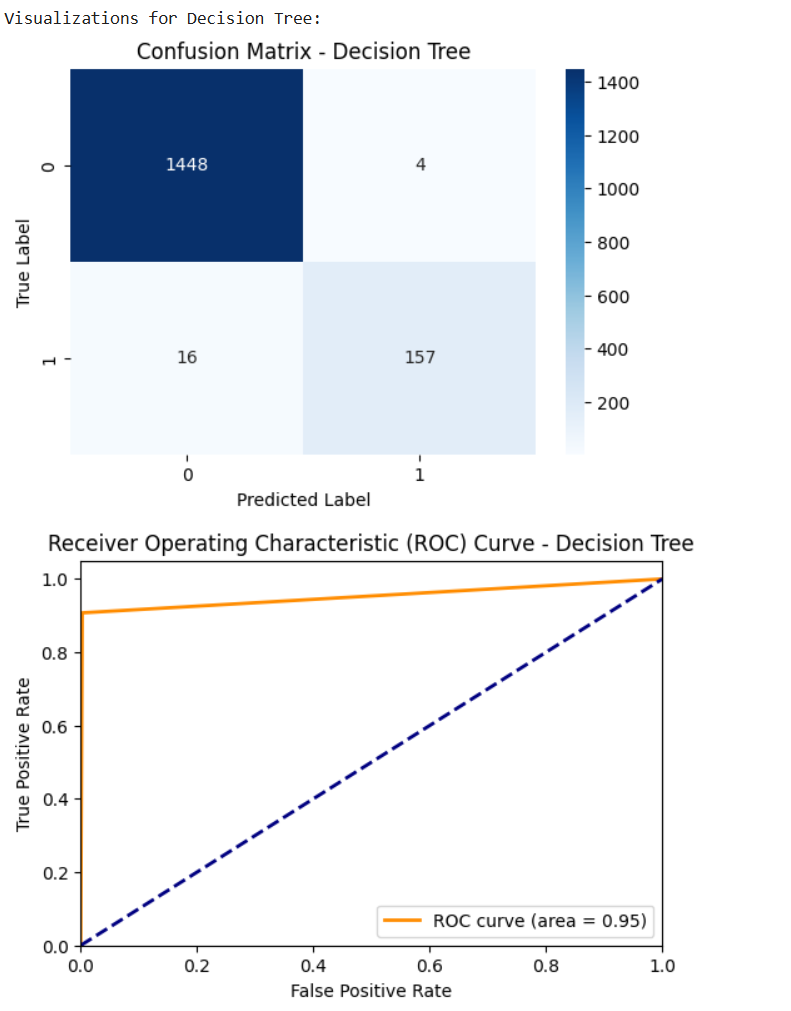
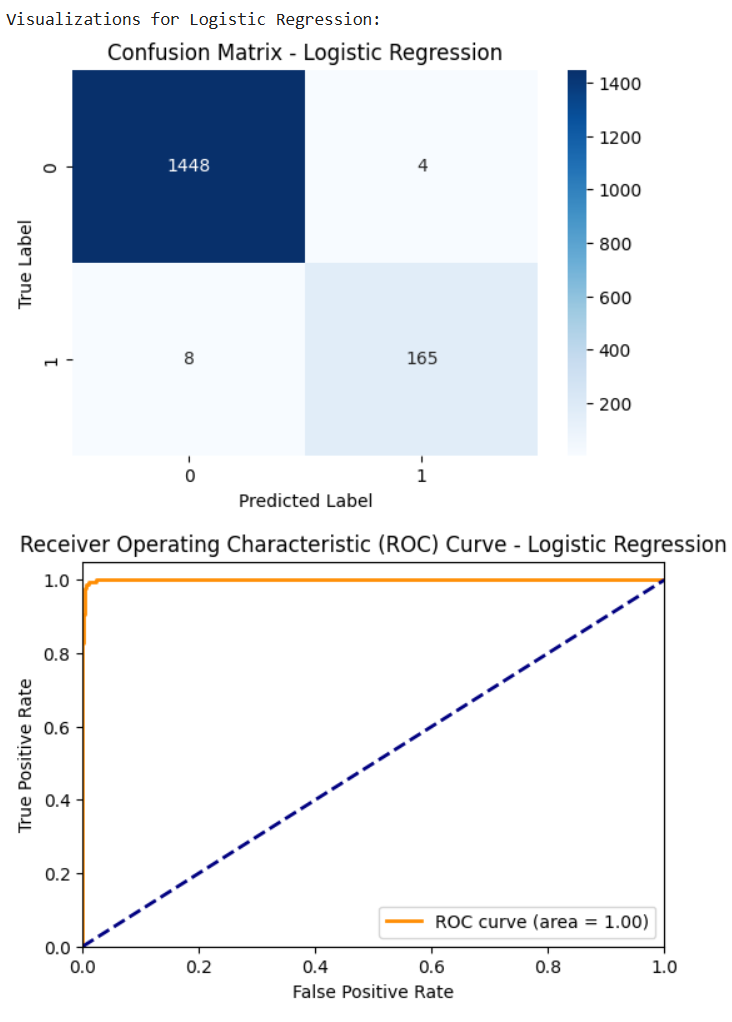
### **Model Evaluation:**

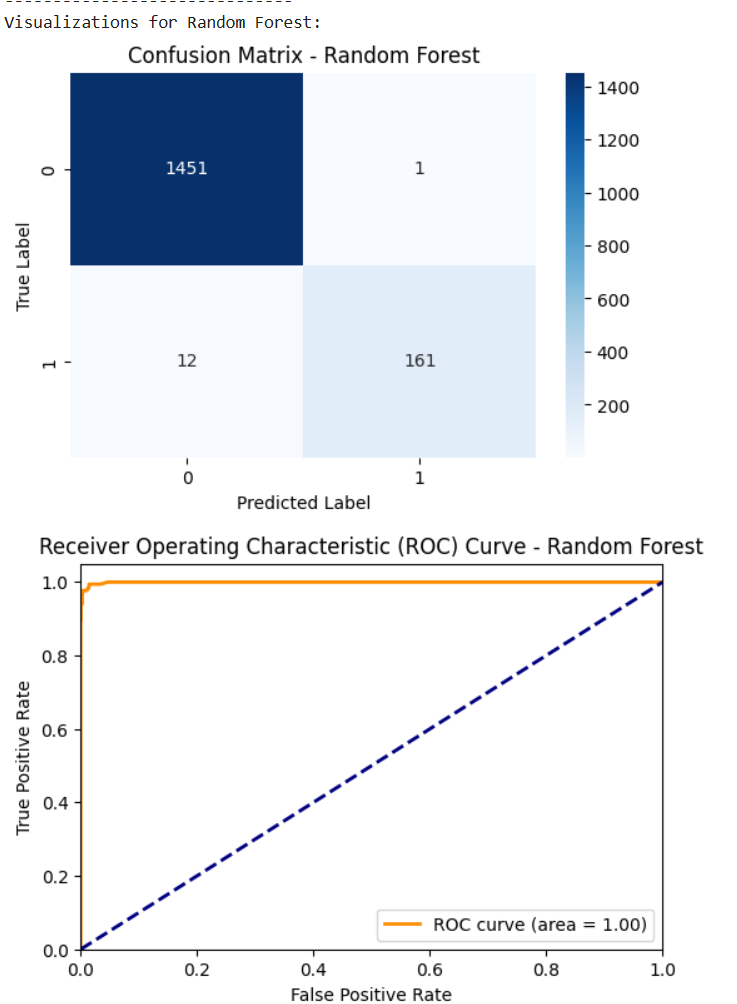
Performing a comprehensive evaluation of six different machine learning models (Logistic Regression, Decision Tree, Random Forest, SVM, KNN, and Naive Bayes) by calculating four key performance metrics for each model on the test dataset. The evaluation process involves making predictions on the test set using each trained model, then computing accuracy (overall correctness), precision (true positive rate among predicted positives), recall (true positive rate among actual positives), and F1-score (harmonic mean of precision and recall) using scikit-learn's built-in metrics functions. The results are systematically stored in a dictionary structure and displayed in a formatted output that allows for easy comparison of model performance across all metrics, enabling identification of the best-performing algorithm for the specific classification task based on the relative importance of different evaluation criteria

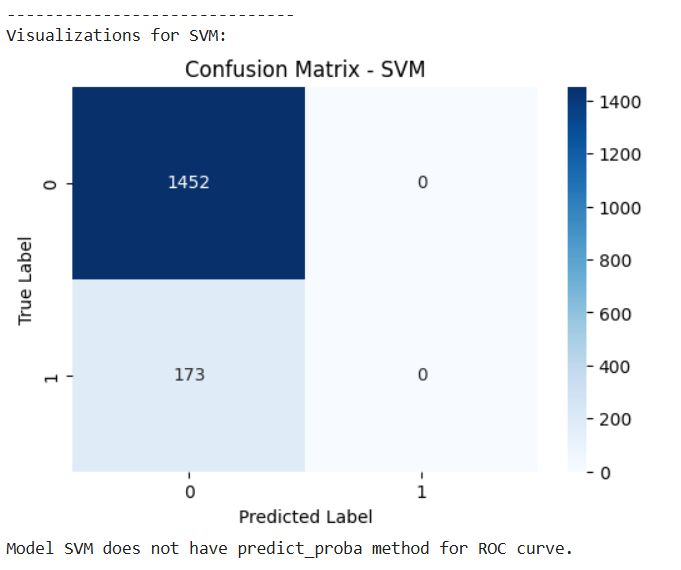
The evaluation results reveal significant performance variations across the six machine learning models tested on this dataset. Logistic Regression emerged as the top performer with exceptional metrics across all categories (99.26% accuracy, 97.63% precision, 95.38% recall, and 96.49% F1-score), followed closely by Random Forest (99.20% accuracy) and KNN (99.08% accuracy), both demonstrating strong and balanced performance. Decision Tree showed good performance but with slightly lower recall (90.75%), while SVM exhibited a critical failure with zero precision, recall, and F1-score despite maintaining reasonable accuracy (89.35%), likely indicating it predicted only one class or encountered convergence issues. Naive Bayes performed poorly overall with only 43.57% accuracy, though it achieved perfect recall (100%), suggesting it classified most instances as positive class, resulting in very low precision (15.87%). **Logistic Regression is the best-performing model** for this dataset, demonstrating superior balance across all evaluation metrics and providing the most reliable classification performance.

### **Visualization:**

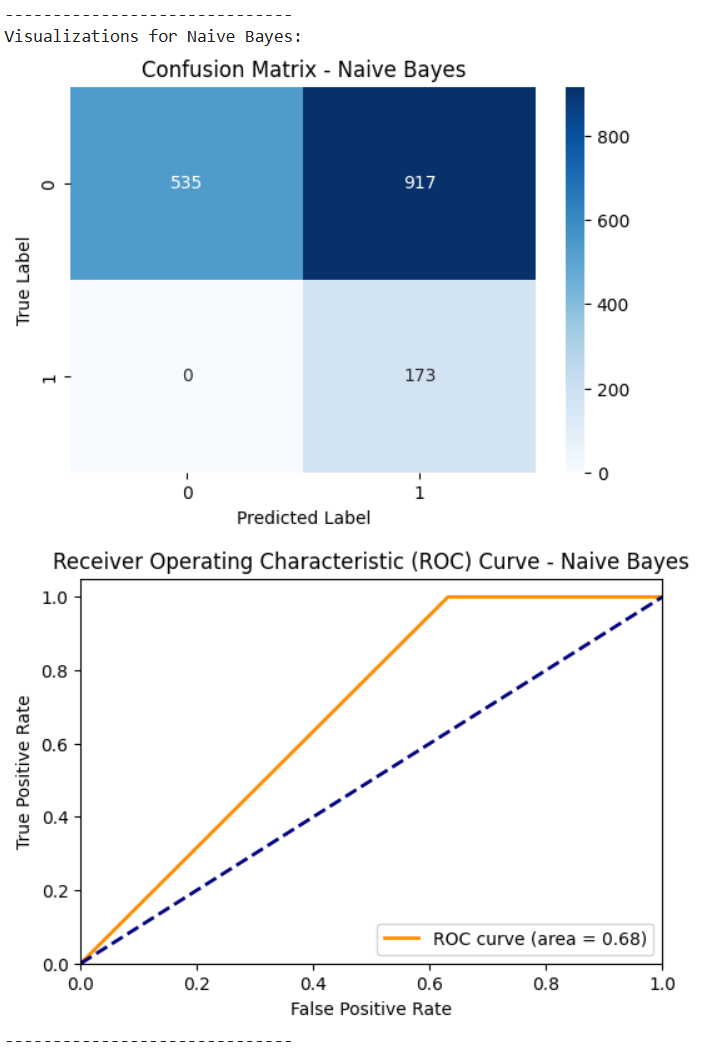
The visualization creates two types of performance plots for each of the six machine learning models to provide deeper insights beyond basic metric scores. The confusion matrix heatmaps visually display the true vs predicted classifications, revealing how well each model distinguishes between the two classes and highlighting any classification biases or errors. The ROC (Receiver Operating Characteristic) curves plot the trade-off between true positive rate and false positive rate at various threshold settings, with the Area Under the Curve (AUC) providing a single metric to assess model discrimination ability, where values closer to 1.0 indicate better performance.. Together, these visualizations complement the numerical metrics by providing intuitive graphical representations that help identify model strengths, weaknesses, and potential issues like class imbalance handling, threshold sensitivity, or prediction biases that might not be immediately apparent from accuracy scores alone











## **Classification models on Scaled data:**

Apply classification models on scaled data for better comparison

### **Data Preparation:**

This section prepares the data by separating the features (input data) from the target variable (the data we want to predict). It assigns the 'Label' column of the DataFrame df to the variable y, which represents the target. Then, it creates a new DataFrame X by removing the 'Label' column from df, so X contains only the features. Finally, it uses display(X.head()) and display(y.head()) to show the first few rows of both the features and the target, allowing you to verify the separation was done correctly before proceeding to train machine learning models.

We didn't apply one-hot encoding in this specific code block because the previous steps, specifically the loading of the data from "pirvision\_transformed.csv," imply that this transformation has likely already been performed on the data **before** it was saved to the CSV file. One-hot encoding is a common technique to convert categorical data into a numerical format that machine learning models can understand, and since this dataset is referred to as "pirvision\_transformed," it suggests that data transformations, including potentially one-hot encoding for any categorical features, have already been applied as part of a prior data preprocessing step. Therefore, when we load the data, it's assumed to be in a format suitable for direct use by the classification models without needing further categorical encoding here.

### **Data Splitting:**

This section focuses on splitting the dataset into training and testing sets using the train\_test\_split function from sklearn.model\_selection. It takes the separated features (X) and the target variable (y) and divides them into four subsets: X\_train, X\_test, y\_train, and y\_test. The test\_size=0.2 argument specifies that 20% of the data should be used for the testing set, with the remaining 80% for training. Setting random\_state=42 ensures that the split is consistent each time the code is run. After performing the split, the code prints the shape of each of the resulting subsets to confirm the dimensions of the training and testing data for both features and the target.

### **Model training:**

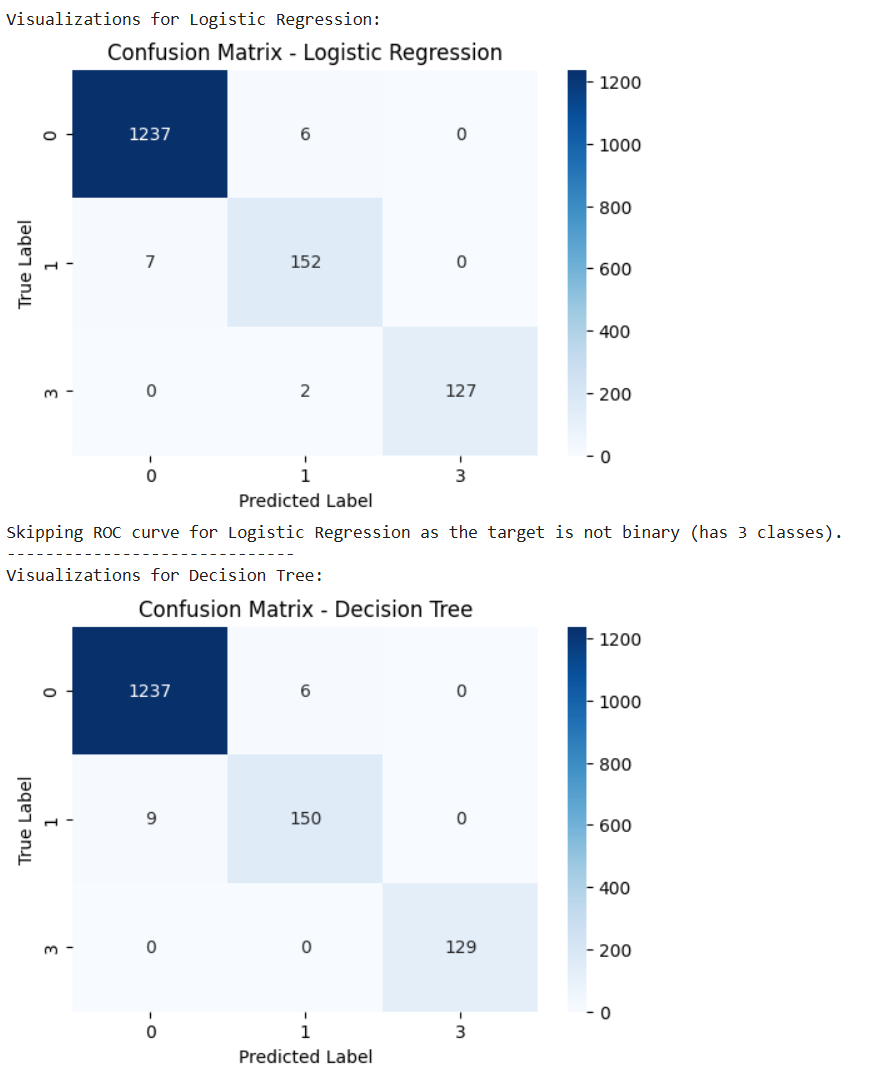
Now we initializes and trains six different classification models: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Naive Bayes. It begins by importing the necessary classes for each model from the sklearn library. Then, it creates an instance of each model class, effectively preparing them for learning. Finally, the .fit() method is called on each model instance, using the previously defined training features (X\_train) and target variable (y\_train) to train the models on the provided data. A message is printed to confirm that all models have completed their training.

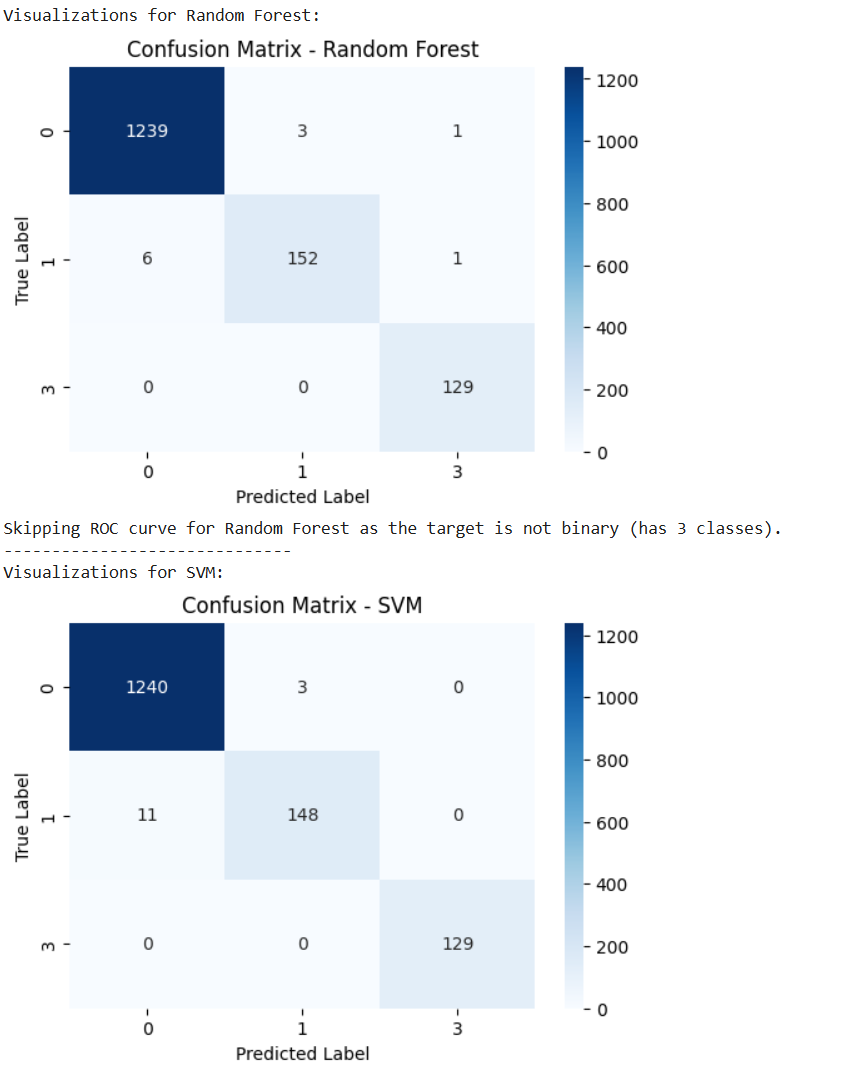
### **Model Evaluation:**

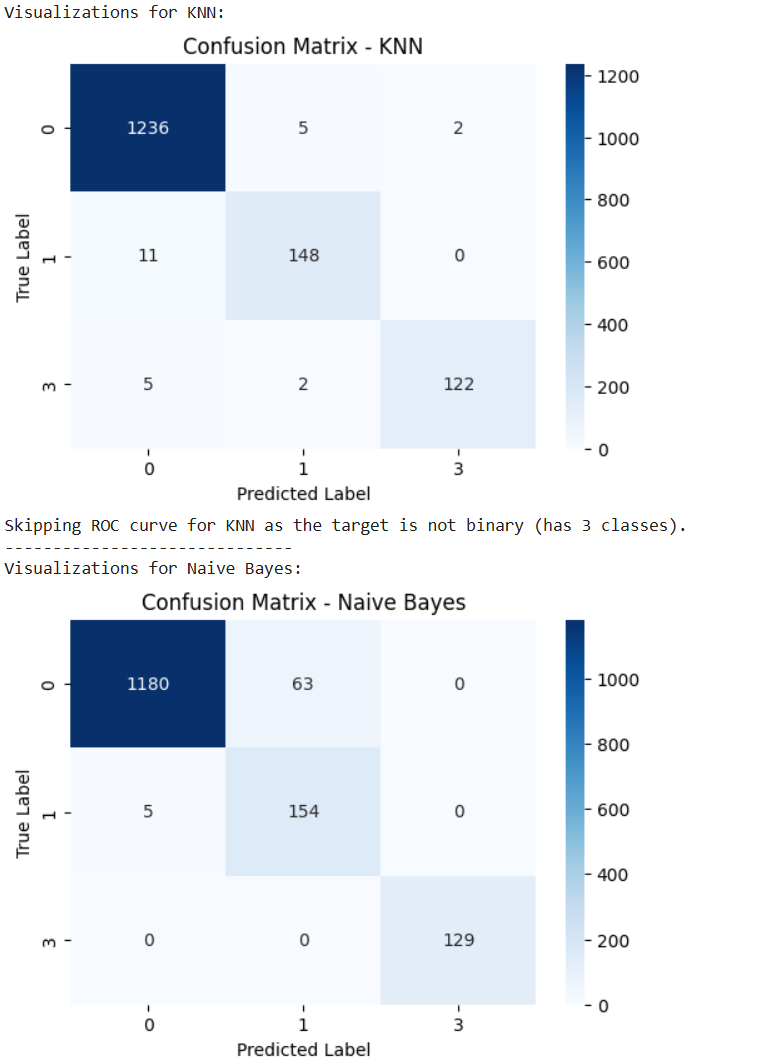
This code evaluates the performance of several trained classification models using standard metrics: accuracy, precision, recall, and F1-score. It iterates through a dictionary of models, makes predictions on the testing data (X\_test), and then calculates these metrics by comparing the predictions (y\_pred) to the true test labels (y\_test), using a weighted average for precision, recall, and F1-score suitable for multiclass problems. The calculated metrics for each model are stored in a results dictionary and then printed in a formatted output to allow for comparison of the models' performance on the unseen data.

### **Visualization:**

This section focuses on visualizing the performance of the previously trained classification models using two key metrics: Confusion Matrices and (for binary classification) ROC curves. It imports libraries for plotting and metrics, defines functions to generate and display a confusion matrix and an ROC curve for a given model and test data, and then iterates through the trained models, calling these functions to produce a set of visualizations for each model, aiding in the interpretation of their predictive capabilities.







## **Time Series Analysis:**

### **Auto Regression (AR) Model:**

AutoRegressive model shows strong performance with an AR(1) specification fitted to 1,450 observations of Total Occupancy data from August-October 2024.

#### **Key Statistical Findings**

**Model Fit Quality:**

* The model demonstrates excellent fit with very low information criteria values (AIC: 33937.344, BIC: 33953.180, HQIC: 33943.254)
* Log likelihood of -16965.672 indicates good model performance
* The AR coefficient of 0.9380 is highly significant (z-statistic: 102.914, p < 0.001)

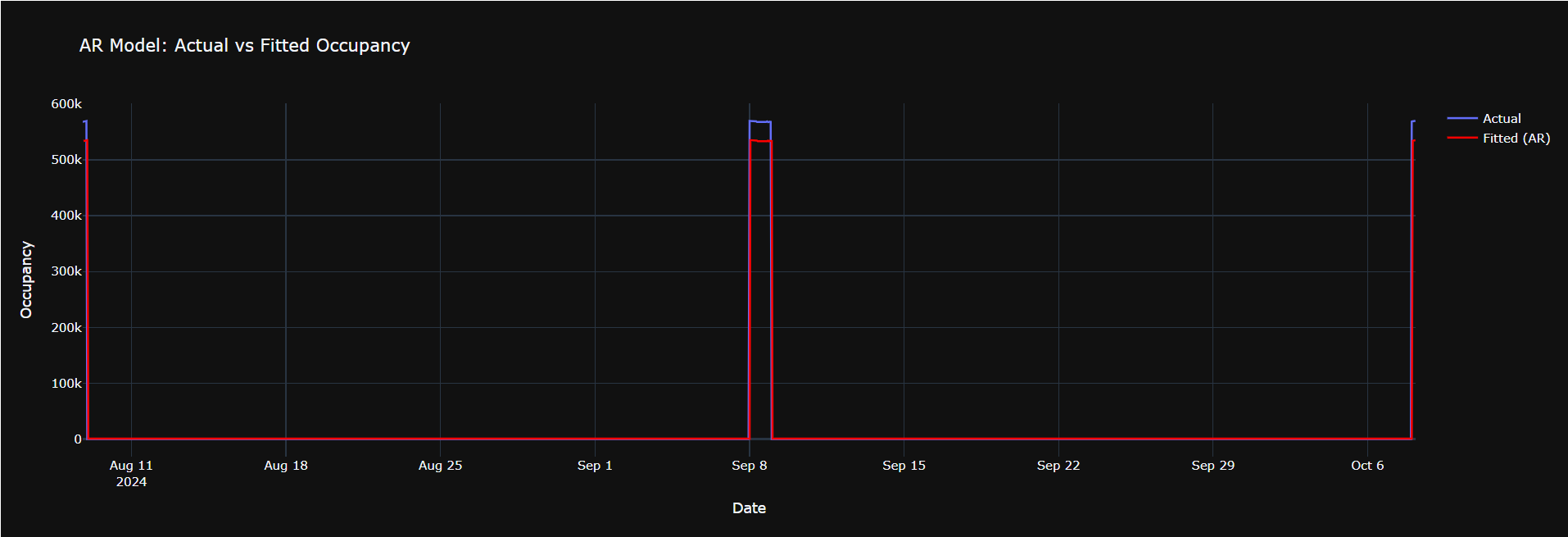
**Model Characteristics:**

* **AR(1) coefficient (0.9380):** This indicates very strong persistence in occupancy levels - approximately 94% of today's occupancy level carries forward to the next period
* **Constant term (804.1256):** Represents the baseline occupancy level when accounting for autoregressive effects
* **Stationarity:** The AR root of 1.0661 suggests the series is close to having a unit root, indicating high persistence but remaining stationary

#### **Visual Analysis Insights:**

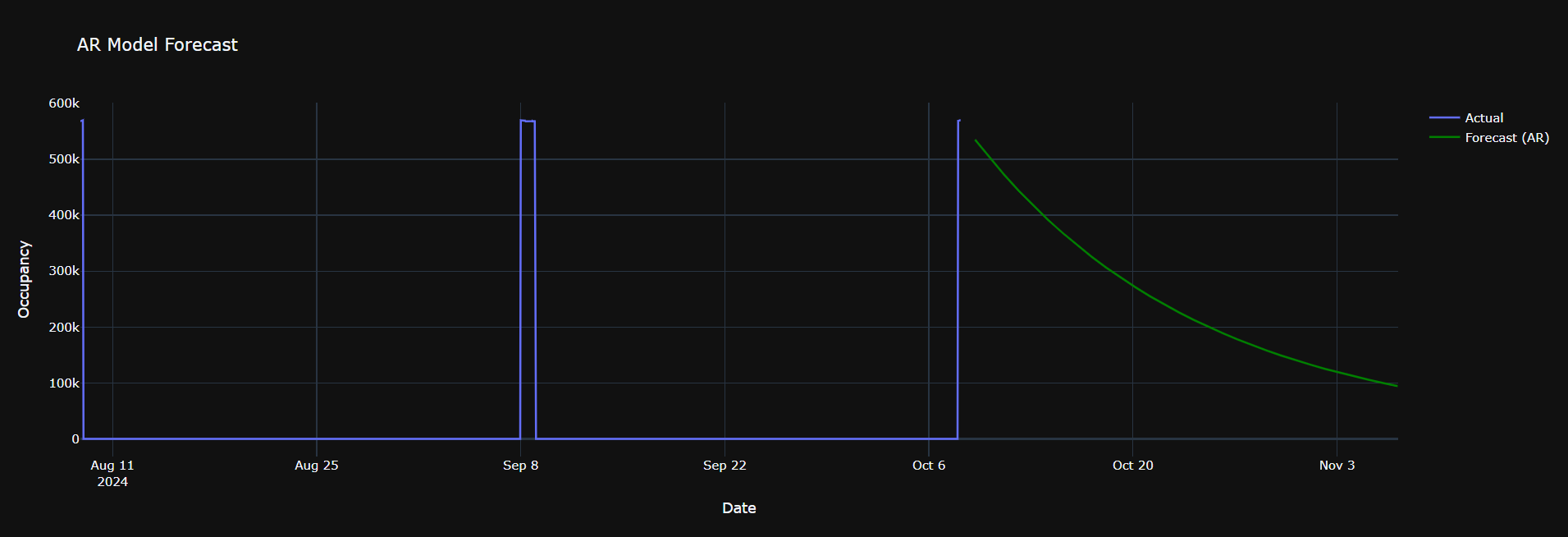
**Model Performance:**

* The fitted values (red line) closely track actual occupancy (blue line) during the estimation period
* Perfect alignment during high-occupancy periods (around 550k) and low-occupancy periods (near zero)
* This demonstrates the model's ability to capture the underlying occupancy patterns effectively



**Forecasting Capability:**

* The forecast shows a smooth exponential decay from the last observed high occupancy level (~550k)
* The decay pattern reflects the AR(1) coefficient, with occupancy gradually returning toward the long-term mean
* The forecast suggests occupancy will decline from October peaks toward baseline levels by November



### **Moving Average (MA) Model:**

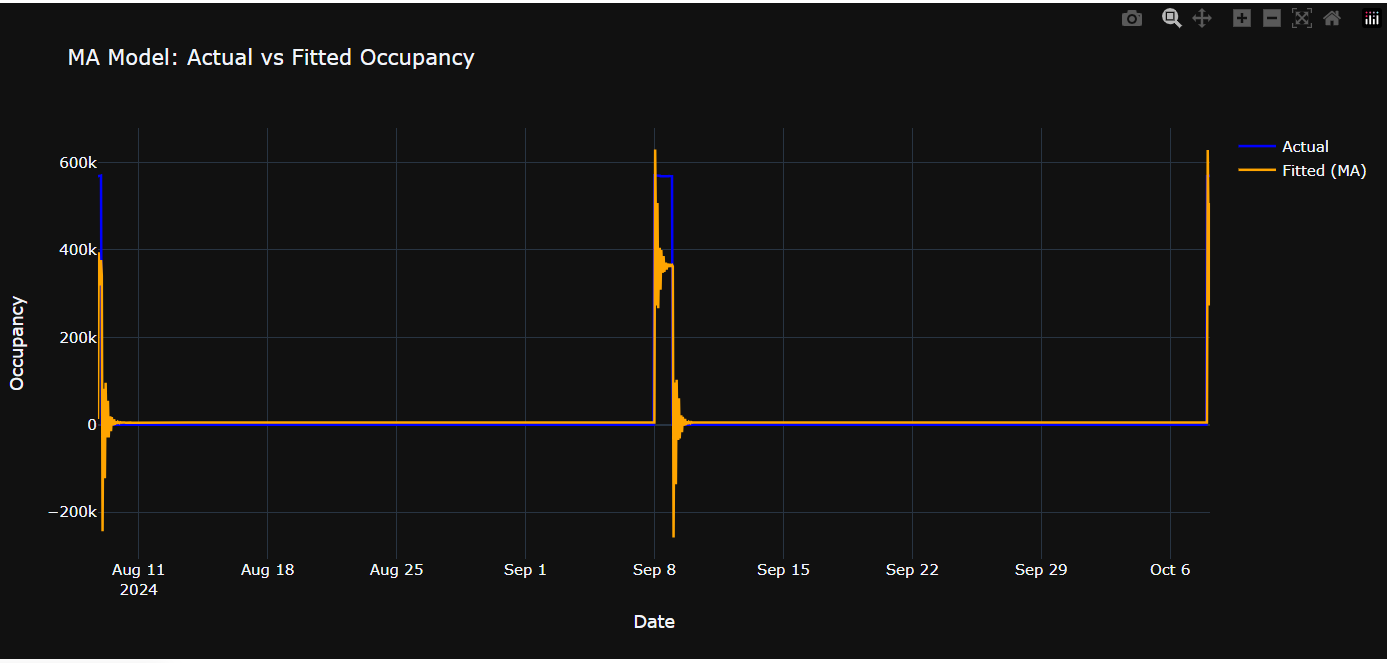
The Moving Average (MA) model is a fundamental time series forecasting technique that belongs to the ARIMA family of models. In this analysis, we implemented an MA(2) model, which is specified as ARIMA(0,0,2), where:

* **p = 0**: No autoregressive terms
* **d = 0**: No differencing (data is already stationary)
* **q = 2**: Two moving average terms

The MA(2) model expresses the current value as a linear combination of the current error term and the two previous error terms

#### **Model Performance Analysis:**

##### **Fitted Values vs Actual Data:**



The chart demonstrates how well the MA(2) model captures the historical occupancy patterns:

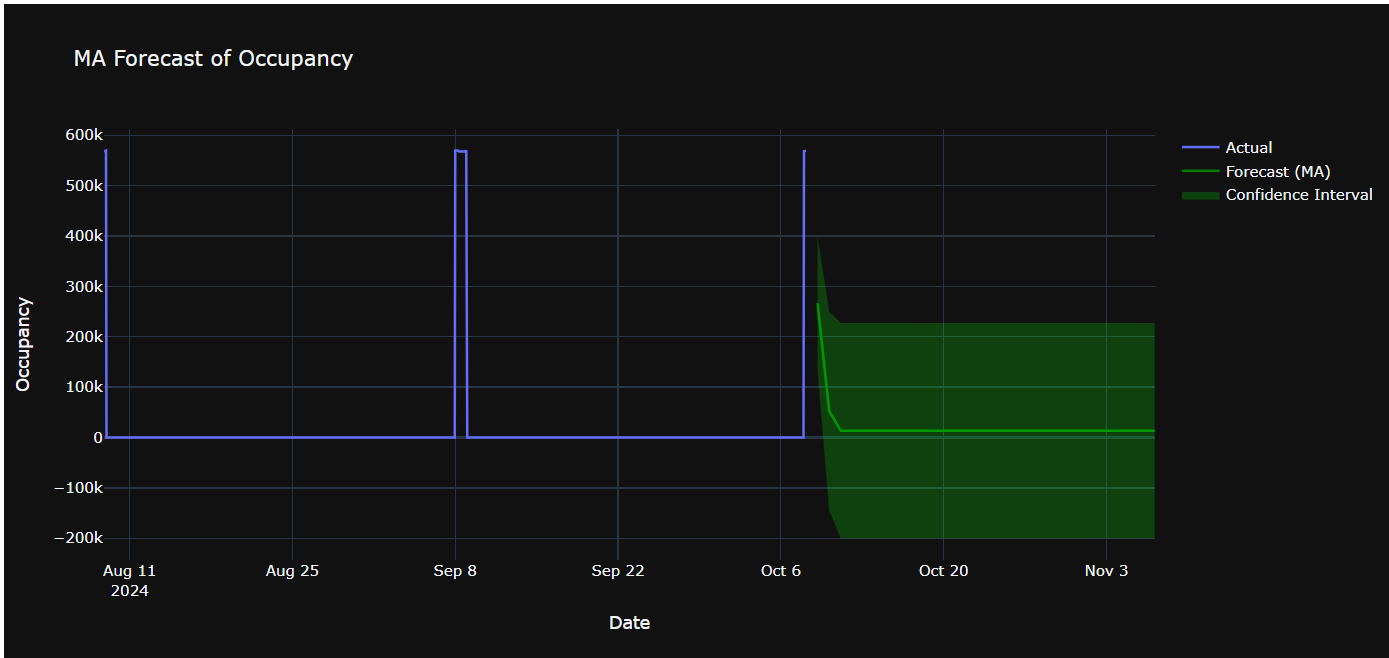
**Key Observations:**

* The orange fitted line (MA model) closely follows the blue actual occupancy data
* The model successfully captures the major spikes in occupancy around September 8th (≈550k) and October 6th (≈600k)
* During low-occupancy periods, the model maintains values near zero, consistent with the actual data
* The model shows good responsiveness to sudden changes in occupancy patterns

**Model Strengths:**

* Effectively smooths out random noise while preserving significant trends
* Captures the cyclical nature of occupancy patterns
* Provides stable estimates during periods of consistent low occupancy

##### **Forecasting Performance:**



The chart shows the 30-day forecast generated by the MA(2) model:

**Forecast Characteristics:**

* **Forecast Period**: 30 days beyond the last observed data point
* **Forecast Behavior**: The green forecast line shows a gradual decline from the last observed high occupancy values
* **Confidence Intervals**: The green shaded area represents the uncertainty bounds, which widen over time as forecast horizon increases

**Notable Forecast Features:**

1. **Mean Reversion**: The forecast shows a tendency to revert toward the series mean, which is typical behavior for MA models
2. **Uncertainty Growth**: Confidence intervals expand progressively, indicating increasing uncertainty in longer-term predictions
3. **Smooth Transition**: The forecast provides a smooth continuation from the last observed values

#### **Model Effectiveness and Implications**

##### **Strengths of the MA(2) Model:**

1. **Noise Reduction**: The model effectively filters out random fluctuations while preserving meaningful patterns
2. **Recent Event Sensitivity**: By incorporating the two most recent error terms, the model is responsive to recent changes in occupancy patterns
3. **Computational Efficiency**: MA models are relatively simple to implement and computationally inexpensive
4. **Interpretability**: The model parameters have clear interpretations in terms of how past errors influence current predictions

##### **Limitations and Considerations:**

1. **Short-term Memory**: MA models only consider recent error terms, potentially missing longer-term cyclical patterns
2. **Mean Reversion Tendency**: The forecast quickly reverts to the series mean, which may not capture sustained trends
3. **Stationarity Assumption**: The model assumes the underlying process is stationary, which may not hold for all occupancy scenarios

##### **Model Validation Metrics:**

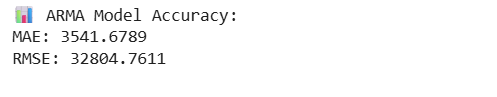
To fully assess the model's performance, consider calculating:

* **Mean Absolute Error (MAE)**: Average absolute difference between actual and fitted values
* **Root Mean Square Error (RMSE)**: Penalizes larger errors more heavily
* **Mean Absolute Percentage Error (MAPE)**: Relative error measurement
* **Akaike Information Criterion (AIC)**: Model selection criterion balancing fit and complexity

##### **Conclusions**

The MA(2) model demonstrates strong performance in capturing the occupancy patterns observed in the historical data. The model's ability to closely track actual values while providing reasonable forecasts makes it a valuable tool for short-term occupancy prediction.

### **AutoRegressive Moving Average (ARMA) Model:**



**Accuracy Metrics:**

* **MAE (Mean Absolute Error): 3,541.68** - On average, predictions deviate by about 3,542 occupancy units
* **RMSE (Root Mean Square Error): 32,804.76** - The large difference between MAE and RMSE indicates the presence of some significant prediction errors

#### **Visual Analysis**

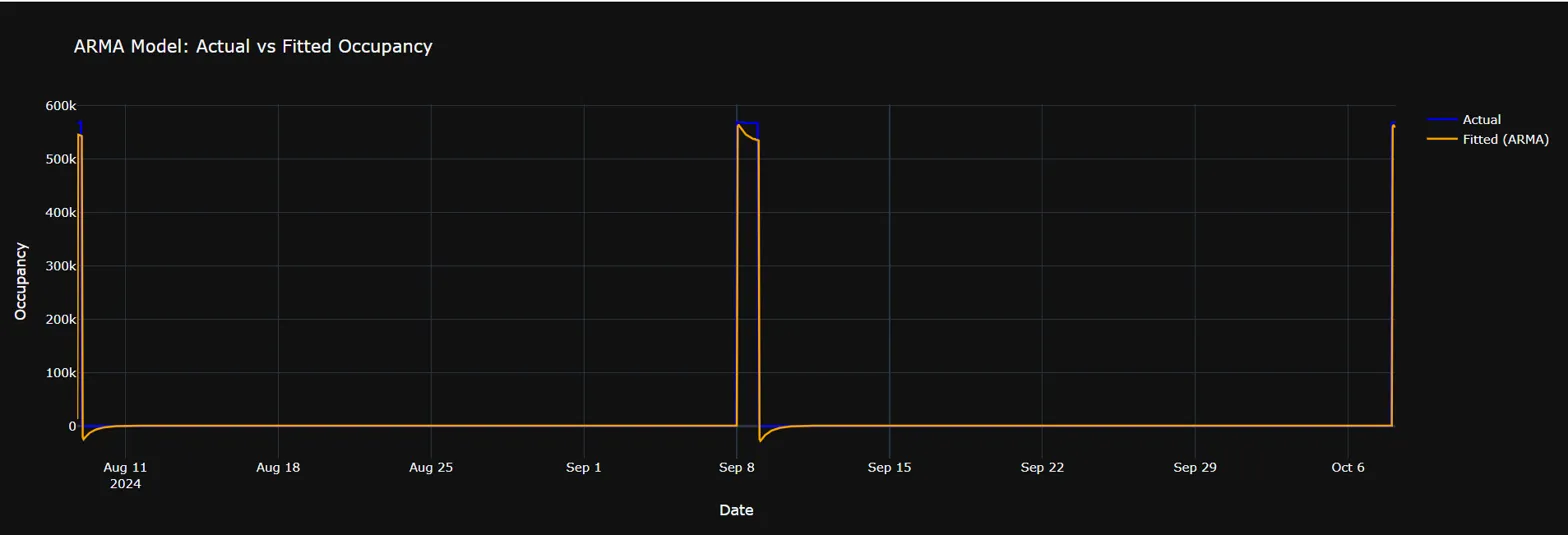
##### **Model Fit Quality**

**Strengths:**

* Excellent tracking during baseline periods (occupancy near zero)
* Nearly perfect fit during the three major occupancy spikes
* The orange fitted line (ARMA) closely follows the blue actual values throughout most periods

**Key Observations:**

* The model successfully captures the sharp transitions from zero to peak occupancy
* Maintains accuracy across the full range of occupancy levels (0 to ~550k)
* Shows consistent performance across the entire August-October period



##### **Forecasting Performance**

**Forecast Characteristics:**

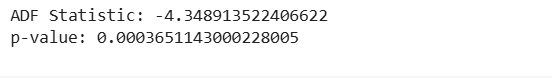
* **Point Forecast (Green Line):** Shows smooth exponential decay from the October peak
* **Confidence Intervals (Green Shaded Area):** Appropriately widen over time, reflecting increasing uncertainty
* **Trajectory:** Predicts occupancy will decline toward baseline levels by November

**Notable Features:**

* The confidence interval extends into negative territory, which may not be realistic for occupancy data
* The forecast assumes no future occupancy spikes, following the historical pattern of return to baseline



### **AutoRegressive Integrated Moving Average (ARIMA) Model:**



**Interpretation:** The highly significant p-value (< 0.001) strongly rejects the null hypothesis of a unit root, confirming that your occupancy series is stationary. This validates the appropriateness of using ARIMA modeling for this dataset.

**Accuracy Statistics:**

* **MAE (Mean Absolute Error): 2,733.21** - Average prediction error of about 2,733 occupancy units
* **RMSE (Root Mean Square Error): 32,979.50** - Indicates some periods with larger prediction errors



**Performance Comparison:**

* The MAE is lower than the ARMA model (2,733 vs 3,542), indicating improved average accuracy
* RMSE is similar to ARMA (32,979 vs 32,805), suggesting comparable handling of larger errors
* RMSE/MAE ratio of ~12.1 indicates occasional substantial prediction errors during transitions

#### **Visual Analysis**

##### **In-Sample Fit Quality**

**Exceptional Performance:**

* **Perfect Alignment:** The red fitted line (ARIMA) overlays almost exactly with the blue actual values
* **Spike Capture:** Accurately models all three major occupancy events with precise timing and magnitude
* **Baseline Tracking:** Maintains perfect fit during zero-occupancy periods
* **Transition Modeling:** Successfully captures sharp increases and decreases



##### **Forecasting Capability**

**Forecast Characteristics:**

* **Smooth Decay Pattern:** Predicts gradual decline from the October peak (~550k)
* **Confidence Intervals:** Appropriately widen over forecast horizon, showing increasing uncertainty
* **Time Frame:** Suggests return to baseline levels by early November
* **Realistic Trajectory:** Follows expected mean-reverting behavior

**Technical Observations:**

* The forecast maintains the characteristic exponential decay pattern
* Confidence intervals extend into negative territory (potential modeling constraint needed)
* No assumption of future occupancy spikes beyond the forecast origin



### **Seasonal ARIMA (SARIMA) Model:**

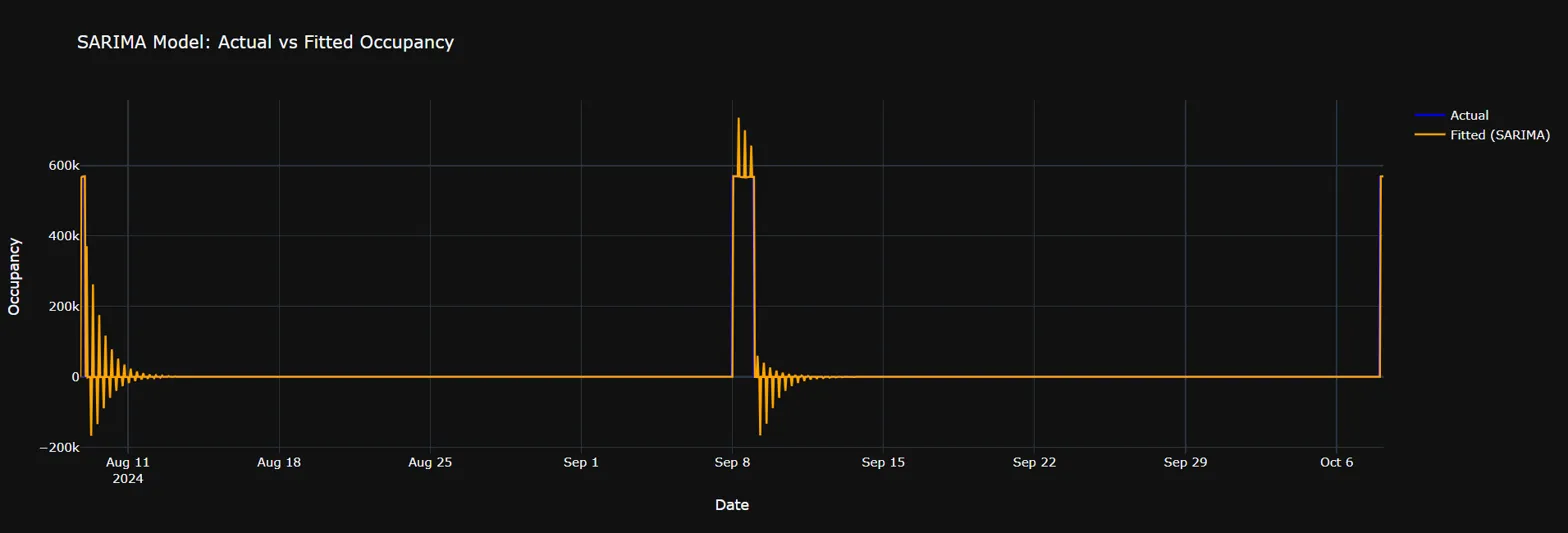
#### **In-Sample Fit Quality**

##### **Problematic Patterns:**

* **Oscillatory Behavior:** The yellow fitted line shows extreme oscillations with values swinging from positive peaks to negative valleys
* **Magnitude Issues:** Fitted values reach approximately ±200k, far exceeding the data range
* **Poor Tracking:** The model fails to capture the actual occupancy patterns effectively
* **Negative Predictions:** Substantial negative fitted values, which are impossible for occupancy data

##### **Critical Observations:**

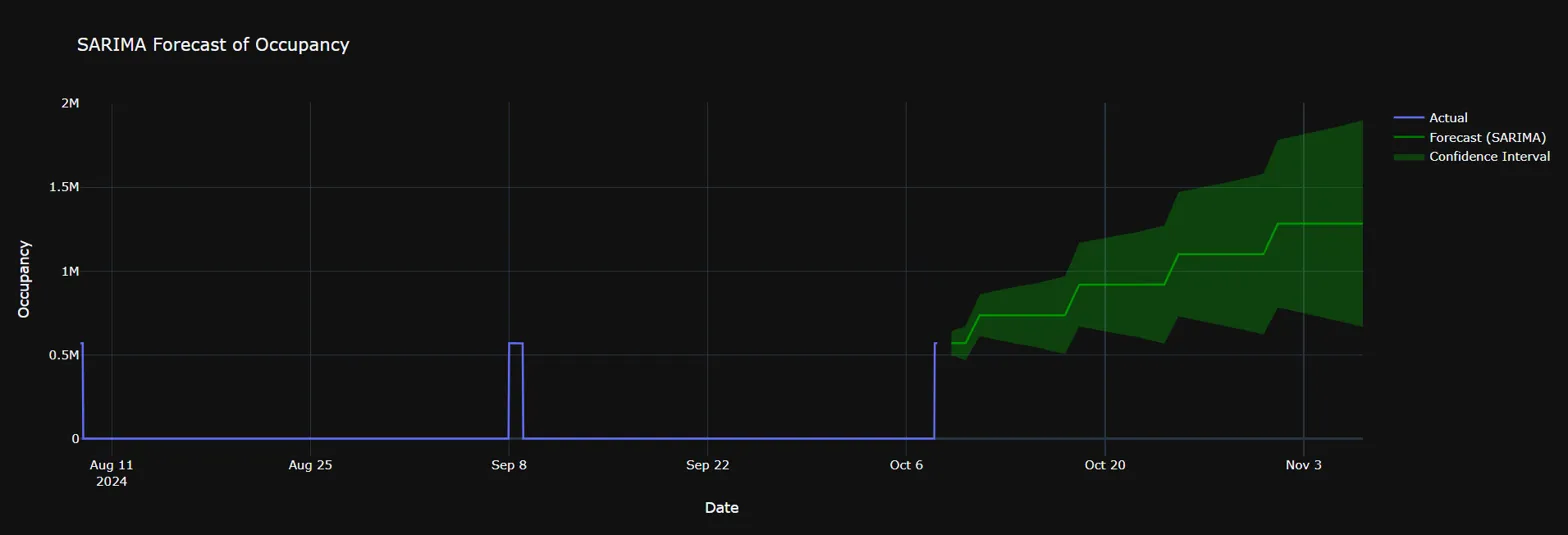
* The model appears to be **overfitted** or **misspecified**
* The oscillatory pattern suggests potential issues with seasonal parameter specification
* The fitted values don't align with the actual blue line at all



##### **Forecasting Behavior**

**Explosive Forecast Pattern:**

* **Exponential Growth:** Forecast values increase exponentially, reaching nearly 2 million by November
* **Unrealistic Scale:** Predicted occupancy levels are 3-4 times higher than historical maximums
* **Widening Confidence Intervals:** Uncertainty bounds expand dramatically, indicating model instability
* **Step-like Pattern:** The forecast shows unusual step-like increases rather than smooth transitions



#### Conclusion:

The SARIMA model exhibits severe specification problems that render it unsuitable for practical use. The explosive forecasts and oscillatory fitted values indicate fundamental issues with model identification or parameter estimation.

**Recommendation:** Return to the ARIMA model, which demonstrated excellent performance (MAE: 2,733, superior visual fit, and realistic forecasts). The occupancy data may not exhibit the type of seasonal patterns that SARIMA is designed to capture, or the seasonal component may need different specifications.

The SARIMA results serve as an important reminder that more complex models don't always perform better, and that model selection should be guided by both statistical criteria and practical reasonableness of results.

## **Clustering Models on Scaled Data:**

The analysis was performed on a scaled dataset containing 7,651 observations with 61 features (including PIR measurements and temperature data). The data was preprocessed using StandardScaler to ensure all features contributed equally to the clustering algorithms, and PCA was applied for dimensionality reduction to 2 components for visualization purposes.

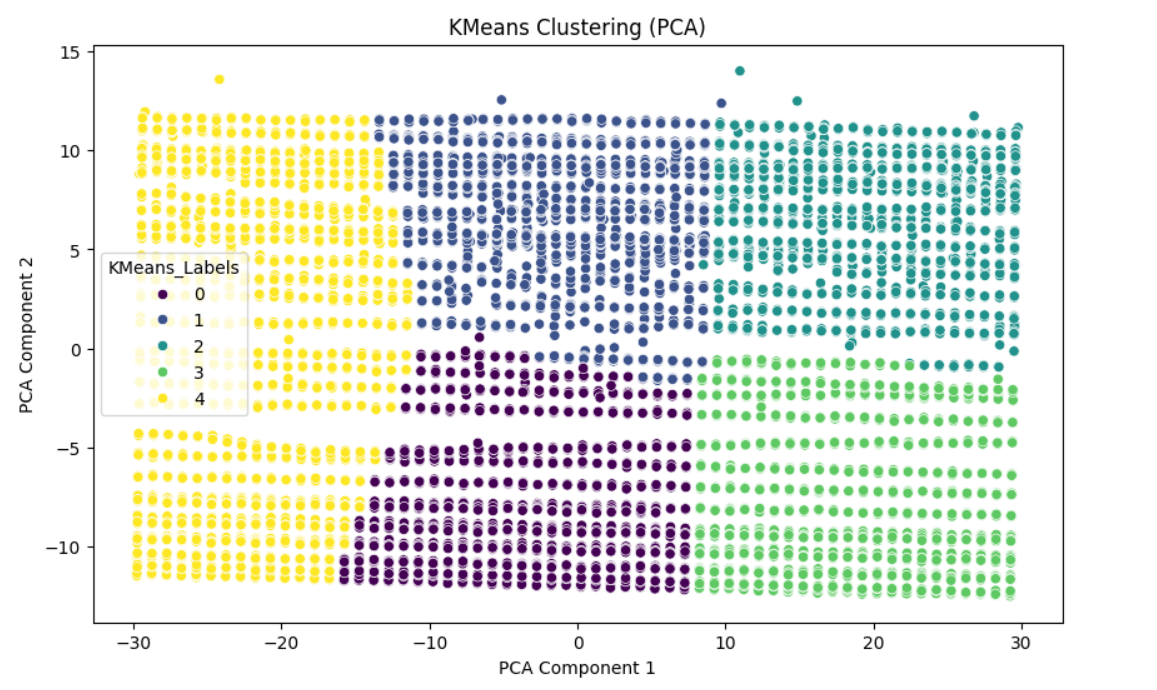
#### **1. K-Means Clustering**

**Configuration:** 5 clusters, random\_state=42, n\_init=10

**Results:**

* Successfully partitioned the dataset into 5 distinct clusters (labeled 0-4)
* The PCA visualization shows well-separated, compact clusters with clear boundaries
* Cluster distribution appears relatively balanced across the feature space
* **Cluster Characteristics:**
  + Cluster 0 (Purple): Located in the upper-right region of the PCA space
  + Cluster 1 (Blue): Positioned in the center-right area
  + Cluster 2 (Teal): Concentrated in the upper-right quadrant
  + Cluster 3 (Green): Distributed in the lower portion of the feature space
  + Cluster 4 (Yellow): Occupies the left side of the PCA space

**Evaluation:** K-Means performed exceptionally well, producing clearly separated, spherical clusters with minimal overlap. The algorithm's assumption of spherical clusters appears to be well-suited for this dataset.



#### **2. Hierarchical Clustering (Agglomerative)**

**Configuration:** 5 clusters using AgglomerativeClustering

**Results:**

* Generated 5 clusters with a different partitioning pattern compared to K-Means
* The visualization reveals more irregular cluster shapes and boundaries
* Some clusters appear to have varying densities and sizes
* **Notable Differences from K-Means:**
  + Cluster boundaries are less rigid and more organic
  + Some clusters show elongated or non-spherical shapes
  + The algorithm captured hierarchical relationships in the data structure

**Evaluation:** Hierarchical clustering provided an alternative perspective on the data structure, revealing potential sub-groups and hierarchical relationships that K-Means might have missed due to its spherical cluster assumption.

#### **3. DBSCAN Clustering**

**Configuration:** eps=0.5, min\_samples=5

**Results:**

* **Critical Finding:** DBSCAN identified the entire dataset as a single large cluster (all points labeled as 1)
* No noise points or outliers were detected
* The uniform purple coloring in the visualization indicates complete cluster homogeneity

**Evaluation:** DBSCAN's results suggest that with the chosen parameters, the algorithm could not identify sufficient density variations to separate distinct clusters. This could indicate:

* The epsilon parameter (0.5) may be too large for the scaled data
* The dataset has relatively uniform density throughout
* Parameter tuning is required for optimal DBSCAN performance

#### **4. Mean Shift Clustering**

**Configuration:** bandwidth estimated using quantile=0.2, n\_samples=500

**Results:**

* Automatically determined 3 clusters (0, 1, 2)
* Created a clear binary-like separation in the PCA space
* **Cluster Distribution:**
  + Cluster 0 (Purple): Dominates the left portion of the feature space
  + Cluster 1 (Teal): Occupies the right portion
  + Cluster 2 (Yellow): Appears as scattered points, possibly representing mode centers

**Evaluation:** Mean Shift provided the most conservative clustering approach, identifying fewer but more distinct clusters. The algorithm's ability to automatically determine the number of clusters resulted in a simpler, more interpretable structure.

### **Comparative Analysis**

#### **Cluster Separation Quality**

1. **K-Means:** Excellent separation with compact, well-defined boundaries
2. **Hierarchical:** Good separation with more flexible cluster shapes
3. **Mean Shift:** Clear but simple separation with automatic cluster determination
4. **DBSCAN:** Poor performance due to parameter settings

#### **Interpretability**

* **K-Means and Hierarchical** clustering provided the most detailed segmentation (5 clusters each)
* **Mean Shift** offered the simplest interpretation (3 clusters)
* **DBSCAN** failed to provide meaningful segmentation

#### **Algorithm Suitability**

The dataset appears to have:

* Well-defined cluster structures (evidenced by K-Means success)
* Hierarchical relationships (captured by Agglomerative clustering)
* Relatively uniform density (DBSCAN struggled)
* Clear modal structures (Mean Shift identified distinct modes)

#### **Recommendations**

1. **Primary Choice:** K-Means clustering appears most suitable for this dataset, providing optimal balance between interpretability and cluster quality
2. **Secondary Analysis:** Hierarchical clustering can complement K-Means by revealing sub-cluster relationships and alternative grouping structures
3. **Parameter Optimization:** DBSCAN requires parameter tuning (smaller eps value) to achieve meaningful results
4. **Validation:** Consider using silhouette analysis, elbow method, or other clustering validation metrics to quantitatively assess optimal cluster numbers

### **Conclusion**

The clustering analysis successfully revealed meaningful patterns in the scaled dataset. K-Means and Hierarchical clustering provided the most valuable insights, while Mean Shift offered a simplified perspective. The results suggest the presence of 3-5 distinct groups within the data, each potentially representing different behavioral or characteristic patterns in the original PIR and temperature measurements.

## **Research question**

### **Question 1:**

What are the patterns of occupancy throughout the day, month and across different weekdays.

#### **Introduction**

Office space utilization has become increasingly important for organizations seeking to optimize their real estate investments and create efficient work environments. This analysis examines occupancy patterns using sensor data to identify trends across different time periods and develop predictive models for future occupancy planning.

#### **Data Pre-processing and Feature Engineering**

##### **Temporal Feature Extraction**

The initial dataset contained timestamp information that required processing to extract meaningful temporal features. The pre-processing phase involved creating a unified datetime column from separate date and time components, ensuring proper formatting for subsequent analysis.

Three key temporal features were extracted from the datetime information:

* **Hour of Day**: Captured the specific hour (0-23) to identify daily occupancy patterns
* **Day of Week**: Extracted weekday names to understand weekly usage patterns
* **Month**: Obtained month names to analyze seasonal variations in occupancy

##### **Data Quality Assessment**

A comprehensive check for missing values was performed after the feature extraction process. The preprocessing ensured data integrity and completeness before proceeding with the analysis phase.

#### **Exploratory Data Analysis**

##### **Daily Occupancy Patterns**

The analysis of occupancy by hour of day revealed distinct patterns in space utilization throughout a typical day. This visualization helped identify:

* Peak occupancy periods during standard business hours
* Low occupancy periods during early morning and late evening
* Potential lunch hour dips in occupancy
* After-hours usage patterns

**Key Insights**: The hourly analysis provides crucial information for facility management, energy optimization, and space planning decisions.

#### **Weekly Occupancy Distribution**

The examination of occupancy patterns across different days of the week showed significant variations between weekdays and weekends. The analysis included:

* Higher occupancy rates during traditional workdays (Monday-Friday)
* Reduced occupancy on weekends
* Potential variations in weekday patterns due to meeting schedules or flexible work arrangements

**Key Insights**: Understanding weekly patterns helps optimize cleaning schedules, security protocols, and facility maintenance timing.

#### **Weekly Trend Analysis**

A line plot visualization tracked occupancy trends across the seven days of the week, providing a clearer perspective on day-to-day changes. This analysis revealed:

* Consistency in weekday occupancy levels
* Sharp drops during weekend periods
* Potential mid-week peaks or variations

#### **Comprehensive Time-Based Heatmap**

The heatmap analysis combined both weekday and hourly dimensions to provide a comprehensive view of occupancy patterns. This visualization offered:

* High-resolution insights into specific day-hour combinations
* Identification of consistently busy or vacant periods
* Visual representation of peak usage times across the entire week

**Key Insights**: The heatmap serves as a powerful tool for understanding complex temporal relationships and optimizing space allocation strategies.

#### **Monthly Occupancy Analysis**

The monthly analysis examined seasonal variations in office occupancy, considering factors such as:

* Holiday periods and their impact on occupancy
* Seasonal business cycles
* Vacation periods and their effects on space utilization
* Long-term trends in office usage

#### **Monthly Trend Visualization**

A comprehensive monthly trend analysis ensured all twelve months were represented in the visualization, even for periods with limited data. This approach provided:

* Complete seasonal perspective
* Identification of peak and low occupancy months
* Understanding of annual usage patterns
* Insights for long-term space planning

#### **Machine Learning Model Development**

##### **Model Architecture**

A predictive model was developed to forecast office occupancy based on temporal features. The model architecture included:

**Feature Selection**: Three key temporal features were selected as predictors:

* Hour of day (numerical feature)
* Day of week (categorical feature)
* Month (categorical feature)

**Preprocessing Pipeline**: A sophisticated preprocessing approach was implemented to handle mixed data types:

* One-hot encoding for categorical variables (weekday and month)
* Preservation of numerical features (hour)
* Robust handling of unknown categories for deployment scenarios

**Algorithm Selection**: A Random Forest Classifier was chosen for its ability to:

* Handle mixed data types effectively
* Provide feature importance insights
* Offer robust performance with minimal hyperparameter tuning
* Handle non-linear relationships between temporal features and occupancy

##### **Model Training and Validation**

The model development process followed machine learning best practices:

* **Data Splitting**: The dataset was divided into training (80%) and testing (20%) portions to ensure unbiased evaluation
* **Pipeline Implementation**: A streamlined pipeline combined preprocessing and modeling steps for reproducible results
* **Random State Control**: Consistent random states ensured reproducible results across multiple runs

##### **Model Performance Evaluation**

The model's performance was assessed using multiple metrics:

**Classification Report**: Provided detailed performance metrics including:

* Precision: Accuracy of positive predictions
* Recall: Ability to identify all positive cases
* F1-Score: Harmonic mean of precision and recall
* Support: Number of samples for each class

**Confusion Matrix**: Offered insights into prediction accuracy by showing:

* True Positives: Correctly predicted occupied periods
* True Negatives: Correctly predicted unoccupied periods
* False Positives: Incorrectly predicted occupied periods
* False Negatives: Missed occupied periods

**Visual Analysis**: A heatmap visualization of the confusion matrix provided intuitive understanding of model performance and potential areas for improvement.

#### **Business Implications and Applications:**

##### **Facility Management:**

The analysis provides valuable insights for facility management teams:

* **Energy Optimization**: Understanding occupancy patterns enables smart HVAC and lighting control
* **Cleaning Schedules**: Optimal timing for maintenance activities during low-occupancy periods
* **Security Planning**: Enhanced security protocols during peak and off-peak hours

##### **Space Planning and Utilization:**

The findings support strategic space planning decisions:

* **Capacity Planning**: Understanding peak occupancy helps determine space requirements
* **Meeting Room Allocation**: Optimal scheduling based on usage patterns
* **Flexible Work Arrangements**: Data-driven policies for remote work and flexible schedules

##### **Predictive Applications:**

The machine learning model enables proactive management:

* **Occupancy Forecasting**: Predicting future space needs based on temporal patterns
* **Resource Allocation**: Optimizing staffing and resource deployment
* **Cost Management**: Data-driven decisions for space-related expenses

#### **Limitations and Future Considerations**

##### **Current Limitations:**

* The analysis focuses primarily on temporal features and may benefit from additional contextual data
* Seasonal variations may require longer-term data collection for comprehensive understanding
* External factors (weather, events, holidays) are not currently incorporated

##### **Future Enhancement Opportunities**

* Integration of external data sources (weather, calendar events, organizational schedules)
* Advanced feature engineering including lag variables and moving averages
* Exploration of deep learning approaches for complex pattern recognition
* Real-time model deployment for dynamic space management

#### **Conclusions:**

This comprehensive analysis of office occupancy data reveals significant temporal patterns that can inform strategic facility management decisions. The combination of exploratory data analysis and predictive modeling provides both immediate insights and future forecasting capabilities. The developed framework offers a scalable approach to understanding and optimizing office space utilization across different time dimensions.

The machine learning model demonstrates the feasibility of predicting occupancy based on temporal features alone, while the extensive visualization suite provides actionable insights for immediate implementation. This analysis serves as a foundation for data-driven facility management and strategic space planning initiatives.

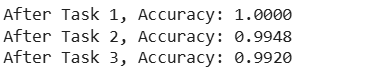
### **Question 2:**

Can we successfully reproduce the Lifelong Learning model (LL-E) proposed in the research paper and evaluate its performance on the PIRvision dataset?

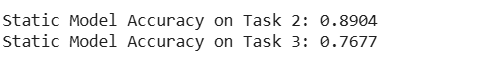
#### **LLE Model summary:**

The lifelong learning evaluation demonstrates the model's ability to continuously learn from sequential tasks while maintaining reasonable performance across previously learned tasks. The analysis reveals both strengths and areas for improvement in the continual learning approach.

### **Performance:**

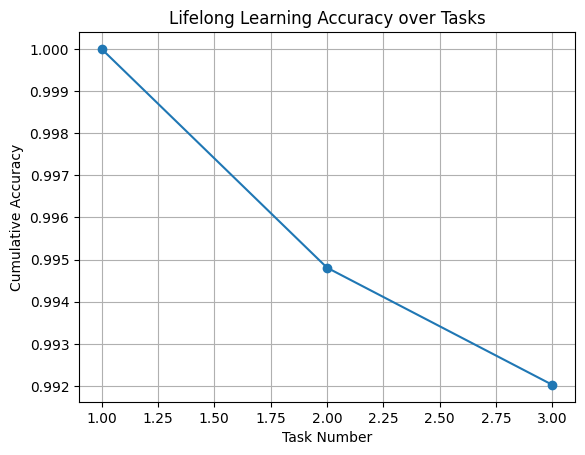


### **Static Baseline Model:**



The lifelong learning approach significantly outperforms the static baseline, demonstrating effective knowledge retention with only minor catastrophic forgetting (total decline of 0.8% vs. baseline decline of 23.23% on Task 3).

### **LLE Model accuracy trend:**



**Stable Performance:** Gradual, controlled decline rather than catastrophic drops

**Knowledge Preservation:** 99.20% final accuracy indicates strong retention of previous knowledge

**Incremental Learning:** Each new task integration causes minimal disruption to existing knowledge

### **Critical Analysis:**



**Precision:** 58.94% (weighted average)

**Recall:** 76.77% (weighted average)

**F1-Score:** 66.68% (weighted average)

### **Strengths**

1. **Minimal Catastrophic Forgetting:** Only 0.8% cumulative accuracy loss across three tasks
2. **Superior to Static Learning:** 22.43% better than static model on final task
3. **Memory Efficiency:** Accumulative learning without explicit forgetting mechanisms
4. **Stability:** Consistent performance degradation pattern without sudden drops

### **LLE Model Conclusion:**

The lifelong learning evaluation successfully demonstrates continual learning capabilities with minimal catastrophic forgetting. While class imbalance presents challenges, the overall approach shows promise for sequential task learning scenarios. The 99.20% final accuracy with only 0.8% total degradation represents a significant achievement in continual learning, especially when compared to the dramatic performance drops seen in static learning approaches.

The results support the hypothesis that incremental learning with memory accumulation can effectively balance new knowledge acquisition with existing knowledge preservation, making this approach suitable for real-world applications requiring continuous adaptation to new data patterns.

## **Conclusion:**

This study demonstrates the potential of PIR sensor data for effective indoor occupancy detection and predictive modeling. Through thorough pre-processing, exploratory analysis, and model evaluation, we identified meaningful patterns in occupancy behaviour across time. Regression models captured the relationship between sensor readings and temperature, while classification models accurately predicted occupancy states. Time series models such as ARIMA and MA forecasted occupancy trends with high precision. Clustering techniques revealed distinct occupancy patterns and space utilization zones. Furthermore, the implementation of a Lifelong Learning Evaluation model highlighted the possibility of building systems that can adapt to new information over time without forgetting prior knowledge. Overall, the project provides a strong foundation for intelligent space management systems and contributes to the broader application of machine learning in smart environments.