SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

ECOSTRUXTURE POWER-THERMAL MONITORING SYSTEM

INT 400 – INTERNSHIP 3

PROJECT REPORT

Submitted by

SUDHARSHINI R – E0322019

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence and Data Analytics)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116

OCTOBER 2024

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

ECOSTRUXTURE POWER -THERMAL MONITORING SYSTEM

INT 400 – INTERNSHIP 3

PROJECT REPORT

Submitted by

SUDHARSHINI R – E0322019

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence and Data Analytics)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116

OCTOBER 2024

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this project report **"ECOSTRUXTURE POWER -THERMAL MONITORING SYSTEM"** is the bonafide record of work done by **"SUDHARSHINI R – E0322019"** who carried out the internship work under my supervision.

Signature of the Supervisor

Dr.B.Nirmala
Associate Professor,
Department of Artificial Intelligence and Data Analytics
Sri Ramachandra Faculty of Engineering and Technology,
SRIHER, Porur, Chennai-600 116.

Signature of Programme Coordinator

Dr. Uma Satya Ranjan
Professor and Head,
Department of Artificial Intelligence and Data Analytics
Sri Ramachandra Faculty of Engineering and Technology,
SRIHER, Porur, Chennai-600 116.

Evaluation Date:

ACKNOWLEDGEMENT

I express my sincere gratitude to our Dean of Sri Ramachandra Faculty of Engineering and Technology, **Dr.T. Ragunathan** for providing the required facilities for this study. I would also like to extend heartfelt thanks to our Head of the Department **Dr. Uma Satya Rajan** for helping us with additional support during this study.

I wish to thank my faculty supervisor(s), **Dr. Nirmala**, Department of Artificial Intelligence and data Analytics, Sri Ramachandra Faculty of Engineering and Technology and **Ms. Ghowsia Samrin Azeez**, Reporting Manager, Schneider Electric for extending help and encouragement throughout the project. Without his/her continuous guidance and persistent help, this project would not have been a success for me.

I am grateful to all the staff members of Sri Ramachandra Faculty of Engineering and Technology, my beloved parents and friends for extending the support, who helped us to overcome obstacles in the study.

.

TABLE OF CONTENTS

TITLE	PAGE NO.
ABSTRACT	**6**
LIST OF TABLES	**7**
LIST OF SYMBOLS	**7**
LIST OF FIGURES	**8**
INTRODUCTION	**9**
1.1 Data Collection	**12**
1.2 Data preprocessing	**12**
1.3 Techniques Involved	**12**
LITERATURE REVIEW	**15**
PROPOSED METHODOLOGY	**16**
IMPLEMENTATION	**17**
RESULTS AND DISCUSSIONS	**21**
APPENDICES	**18**
6.1. Appendix-1: Code – Technical detail	**18**
6.2. Appendix-2: Screenshots	**26**
REFERENCES	**34**
WORKLOG	**37**

ABSTRACT

The Thermal Monitoring System project aims to enhance asset health management by analyzing thermal data from various industrial sites. Data collected from EcoStruxure Asset Advisor, including sensor readings, ambient temperature, and current measurements, is examined for patterns and anomalies around specific thermal events. The analysis spans two weeks before and five days after each event to observe sensor behavior. Visualization techniques, such as time series plots, correlation heatmaps, and data loss frequency analysis, are used to detect abnormalities. Statistical methods, including standard deviation calculation, rolling window analysis, and outlier detection (e.g., Z-score, LOF, One-Class SVM, and K-means), help identify critical deviations and potential faults. The current approach relies on threshold-based flagging, but the project envisions transitioning to unsupervised machine learning models for more accurate and predictive asset health monitoring. Ultimately, this system aims to improve asset reliability, optimize maintenance schedules, and reduce downtime through proactive thermal management.

LIST OF TABLES

S.NO	TABLES	PAGE NUMBER
1.	List of Figures	8
2.	List of Symbols	7
3.	Literature Review	13

LIST OF SYMBOLS

S.NO	ABBREVIATION	EXPANSION
1.	PANDAS	PYTHON DATA ANALYSIS LIBRARY
2.	NUMPY	NUMERICAL PYTHON
3.	SKLEARN	SCIKIT-LEARN
4.	MATPLOTLIB	MATHEMATICAL PLOTTING LIBRARY
5.	PLOTY EXPRESS	GRAPHING LIBRARY

LIST OF FIGURES

S.NO	FIGURE NAME	PAGE NUMBER
1.	Correlation matrix	26
2.	Comparison of the columns	26
3.	Daily frequency of Dataloss	27
4.	Rolling Correlation between S1and CL3	27
5.	Rolling Correlation between S1and CL2	28
6.	Rolling Standard deviation for S2	28
7.	Rolling Standard deviation for CL3	29
8.	Outlier Detection across multiple Sensors using Z-score	29
9.	Comparing Ambient Temperature of different assests	30

CHAPTER 1

INTRODUCTION

The increasing reliance on industrial assets and infrastructure across various sectors necessitates effective monitoring and management to ensure optimal performance and minimize downtime. One of the critical aspects of asset health management is thermal monitoring, as overheating or temperature fluctuations can indicate potential faults or deteriorating conditions. In this context, the Thermal Monitoring System project seeks to analyse and visualize thermal data collected from multiple industrial sites, enabling more efficient predictive maintenance and fault detection.

This project leverages data gathered from EcoStruxure Asset Advisor, a platform that provides real-time monitoring of industrial assets. The platform generates "thermal tickets" when anomalies are detected, and these tickets are associated with specific events or issues in the assets. For each event, the thermal data corresponding to the affected asset is collected, including sensor readings from various thermal and electrical sensors, ambient temperature data, and current readings. To ensure that our analysis captures sufficient context around each event, we gather data for two weeks before and five days after the event date. This time window allows us to examine the asset's thermal behaviour over a period of time and identify any precursors or trends leading up to the event.

Currently, the project focuses on datasets which include multiple assets with distinct operational profiles. These datasets contain several key columns: Date, S1, S2, and S3 (sensor readings), AT (ambient temperature), and CL1, CL2, and CL3 (current readings). By analysing these columns, we can gain insights into how each asset's sensors behave in the lead-up to a fault and detect any anomalies or abnormal fluctuations in their thermal and electrical signatures.

The primary goal of this project is to build a system that can automatically identify, visualize, and analyse thermal anomalies to provide early warnings of potential asset failures. To achieve this, various data analysis techniques are employed. Time series visualizations help in observing the behaviour of sensor data around the event date, allowing for direct comparison of sensor readings before, during, and after the event. We also visualize multiple datasets side-by-side, comparing assets with similar fault conditions to identify patterns or correlations that may suggest common causes.

Additionally, statistical methods are employed to measure and detect fluctuations in the data. These include calculating the standard deviation over time to assess variability, using the rolling window technique to compute rolling correlations and deviations, and performing outlier detection using algorithms such as Local Outlier Factor (LOF), One-Class Support Vector

Machine (SVM), and K-means clustering. These methods help in identifying extreme deviations that could indicate potential faults. The detection of data loss is also a key aspect, as missing or corrupted data can impact the quality of the analysis.

Currently, the system utilizes a threshold-based flagging mechanism, where values exceeding certain thresholds trigger alerts, indicating potential anomalies. This approach is simple but effective in detecting abnormal sensor readings, such as sudden spikes in temperature or current. However, the long-term objective of the project is to transition to unsupervised machine learning models, which will allow for more sophisticated analysis and predictive maintenance capabilities. By employing clustering, anomaly detection, and predictive modelling techniques, the system will be able to provide more accurate assessments of asset health, reducing false alarms and improving the overall reliability of the assets.

In summary, the Thermal Monitoring System project is designed to enhance asset health management by providing detailed insights into thermal behaviour, detecting anomalies, and predicting potential issues before they lead to failures. By leveraging advanced data analysis and machine learning techniques, the project aims to optimize maintenance schedules, reduce unplanned downtime, and improve the overall operational efficiency of industrial assets.

1.1. DATA COLLECTION

Data for the Thermal Monitoring System is collected from EcoStruxure Asset Advisor, focusing on thermal tickets raised for assets at various sites. The datasets include sensor readings (S1, S2, S3), ambient temperature (AT), and current readings (CL1, CL2, CL3), alongside the timestamp (Date) for each entry. Data is gathered for a period spanning two weeks before and five days after each thermal event, providing a comprehensive view of asset behaviour before, during, and after the incident.

1.2. DATA PREPROCESSING

In terms of preprocessing, the first step involves cleaning the data by handling missing values, often through imputation or removal of corrupted entries. Data gaps, which may indicate loss of readings, are flagged for further analysis. Time alignment is essential to synchronize sensor data with the event timestamps, ensuring that data from different sensors or sites is properly aligned. To ensure consistency, features are normalized or standardized, particularly when dealing with sensors that use different units.

1.3. TECHNIQUES INVOLVED

1.2.1. PYCHARM

PyCharm is a powerful Integrated Development Environment (IDE) specifically designed for Python programming. Developed by JetBrains, it offers a rich set of features including intelligent code completion, code inspections, and robust debugging tools. PyCharm facilitates efficient project management and version control integration. It includes tools for database management and testing, enhancing the overall development workflow.

1.2.2. PANDAS

Pandas is an open-source Python library used for data manipulation and analysis. It is especially useful for handling structured data, such as time series data collected from sensors in this project. Pandas offers high-performance data structures like Data Frames that make it easy to clean, filter, and transform large datasets. It is used extensively for data preprocessing, including handling missing values, merging datasets, and performing aggregation operations over time windows.

1.2.3. NUMPY

NumPy is the fundamental package for scientific computing in Python, providing support for large, multi-dimensional arrays and matrices. It offers a variety of mathematical functions that operate on these arrays, making it indispensable for handling numerical data in this project. NumPy is used for operations such as calculating moving averages, rolling window statistics, standard deviation, and other transformations needed for feature engineering and analysis.

1.2.4. SCIKIT-LEARN (SKLEARN)

Scikit-learn is an open-source machine learning library in Python that provides simple and efficient tools for data mining and data analysis. For this project, Scikit-learn is used for anomaly detection and outlier detection using algorithms like Kmeans clustering, One-Class SVM, and Local Outlier Factor (LOF). These techniques help in identifying sensor malfunctions or thermal anomalies that could indicate potential asset failures. Scikit-learn also supports data preprocessing tasks such as scaling and dimensionality reduction.

1.2.5. MATPLOTLIB

Matplotlib is a widely used library for creating static, animated, and interactive visualizations in Python. It is used in this project to create time series plots, histograms, and other visualizations that help explore sensor data over time. Matplotlib is an essential tool for visualizing sensor readings, ambient temperatures, and other variables in relation to the thermal event dates, making it easier to identify patterns and anomalies visually.

.

1.2.6. PLOTLY

Plotly is an interactive data visualization library that supports creating highly customizable and interactive graphs, such as line charts, scatter plots, bar charts, and heatmaps. Plotly is particularly useful for visualizing complex data in a way that allows users to interact with the data through zooming, hovering, and filtering. In this project, Plotly is used to generate interactive plots for analyzing correlations between sensor readings and event occurrences, as well as visualizing data loss and other statistical analysis results.

CHAPTER 2

LITERATURE REVIEW

Survey	Year	Focus Area	Key Findings
Li et al.	**2021**	Infrared Sensors	Effective for non-contact temperature measurement in industrial applications.
Smith & Johnson	**2020**	Thermocouples	Advancements in miniaturization for better integration in smart devices
Jones & Green	**2019**	Thermistors	High sensitivity and accuracy for medical applications, particularly in patient monitoring.
Patel et al.	**2022**	IoT Integration	Real-time monitoring and alerts for temperature deviations in various applications
Nguyen & Tran	**2023**	Machine Learning	Algorithms improve forecasting thermal behaviour in systems, enhancing predictive analytics.
Kumar et al.	**2020**	Industrial Processes	Importance of thermal monitoring to prevent overheating and ensure quality in manufacturing.
Wang et al.	**2021**	Building Management Systems	Automated temperature control enhances energy efficiency and occupant comfort in smart buildings.
Lee et al.	**2022**	Healthcare	Continuous monitoring of patient temperatures for early detection of fever or infection.
Brown & White	**2021**	Calibration and Accuracy	Challenges in ensuring accurate calibration of sensors in harsh environments.
Zhang et al.	**2023**	Data Security	Emphasizes the need for protecting sensitive data in IoT thermal monitoring systems.
Kim et al.	**2024**	Advanced Materials	Research into nanomaterials for higher sensitivity

CHAPTER 3

PROPOSED METHODOLOGY

The proposed methodology for the Thermal Monitoring System involves collecting sensor data from assets at sites using EcoStruxure Asset Advisor. The data includes temperature readings from sensors (S1, S2, S3), ambient temperature (AT), and current readings (CL1, CL2, CL3), collected two weeks before and five days after thermal events. The dataset undergoes preprocessing steps such as cleaning, time synchronization, feature scaling, and handling data loss. To analyze the data, time series visualizations and correlation matrices are used to understand sensor behaviors and interdependencies. Anomaly detection is performed using methods like Z-score outlier detection, Local Outlier Factor (LOF), One-Class SVM, and K-means clustering to identify deviations in sensor readings. The system will further implement predictive models using LSTM (Long Short-Term Memory) networks to forecast future temperature trends based on current sensor data, as well as correlation-based anomaly detection to flag potential faults when there is a significant drift between temperature and current readings. The results of the analysis are visualized using time series plots, and heatmaps to provide actionable insights, enabling proactive maintenance and improved asset health management.

CHAPTER 4

IMPLEMENTATION

The Thermal Monitoring System involves several stages, including data collection, preprocessing, anomaly detection, and predictive modeling, to monitor and analyze the thermal behavior of assets. Below is an overview of how the system is implemented.

1. Data Collection:

Data is collected from the EcoStruxure Asset Advisor platform, which provides real-time monitoring of assets at sites. Each asset generates sensor data, including:

- **Temperature Sensors (S1, S2, S3):** Measuring temperatures at various points of the asset.

- **Ambient Temperature (AT):** The surrounding environmental temperature.

- **Current Sensors (CL1, CL2, CL3):** Monitoring the electrical current flowing through the asset.

- **Date:** Timestamp for each reading, which is essential for time-based analysis.

For each thermal event, data is collected from two weeks before and five days after the event date to understand the asset's behavior before, during, and after the event.

2. Data Preprocessing:

The raw data undergoes several preprocessing steps to ensure it is suitable for analysis:

- **Data Cleaning:** Irrelevant or missing values are handled by either imputing the missing data or discarding rows with insufficient information. Features that do not contribute to the analysis are removed.

- **Time Alignment:** Data is synchronized based on timestamps to ensure the readings from different sensors (temperature, ambient temperature, and current) are aligned with the thermal event dates.

- **Feature Scaling:** Since sensor readings are in different units (e.g., temperature in Celsius and current in amperes), the data is normalized using techniques like Min-Max scaling or Z-score normalization to ensure consistency.

- **Handling Data Loss:** Missing or corrupted sensor readings are tracked, and imputation techniques or removal are applied based on the amount of data loss and its impact on the analysis.

3. Exploratory Data Analysis (EDA):

After preprocessing, Exploratory Data Analysis (EDA) is conducted to uncover insights in the dataset:

- **Time Series Visualization:** Sensor readings are plotted over time, with vertical lines indicating the thermal event date. This visualization allows for comparison of sensor behavior before, during, and after the event.

- **Correlation Analysis:** A correlation matrix and heatmap are generated to understand relationships between different sensors (S1, S2, S3, AT, CL1, CL2, CL3). This helps identify any interdependencies between the asset's performance indicators.

- **Data Loss Frequency:** The frequency and patterns of data loss are analyzed to identify sensors or time periods that are more prone to data loss, which could influence the reliability of future analyses.

4. Anomaly Detection

Anomaly detection is key to identifying abnormal thermal conditions indicative of potential faults. Several techniques are applied:

- **Z-Score Outlier Detection:** Z-scores are computed for each sensor reading, and any value beyond a specified threshold (typically ±3) is flagged as a potential anomaly.

- **Local Outlier Factor (LOF):** LOF detects anomalies by comparing the local density of data points, flagging those that significantly differ from their neighbors.

- **One-Class SVM:** This method identifies regions of the data that deviate from normal patterns, helping to detect new, unseen anomalies.

- **K-means Clustering:** K-means clustering groups data into clusters based on similarities. Data points that do not fit into any cluster are flagged as potential anomalies, highlighting unusual sensor readings.

Predictive Modeling

The system evolves and predicts thermal anomalies before they occur, enabling proactive maintenance. This will involve:

- **Long Short-Term Memory (LSTM) Networks:** LSTM models are used to predict future temperature behavior based on historical current data. By learning patterns in the relationship between temperature and current readings, LSTM will provide accurate temperature forecasts, which can be used to detect potential faults.

- **Correlation-Based Anomaly Detection:** If a significant drift is detected between temperature and current readings, it will be flagged as anomalous. Establishing a strong correlation between these two variables is key to detecting thermal faults early.

- **Rolling Window Analysis:** This technique calculates moving averages and standard deviations for sensor readings over time, helping to detect long-term trends and fluctuations in asset behavior that may signal issues.

6. Visualization

The results are visualized to provide actionable insights:

- Time Series Plots: Time series plots are used to visualize how sensor readings change over time, particularly around thermal events, making it easier to spot significant fluctuations.

- Heatmaps: Correlation heatmaps are used to highlight relationships between sensors, indicating potential dependencies or anomalies.

Conclusion:

The Thermal Monitoring System uses advanced data preprocessing, anomaly detection, and predictive modeling techniques to monitor and predict thermal behavior in industrial assets. The integration of LSTM-based predictive modeling and correlation-based anomaly detection improves the system's ability to forecast potential faults, enabling proactive maintenance.

CHAPTER 5

RESULTS AND DISCUSSIONS

The Thermal Monitoring System successfully detected thermal anomalies in assets using a combination of anomaly detection techniques and predictive analysis. The system effectively identified abnormal thermal behavior using Z-score outlier detection, Local Outlier Factor (LOF), OneClass SVM, and K-means clustering. These methods flagged significant deviations in temperature and current sensor readings, pinpointing potential issues like overheating.

In terms of predictive analysis, the system currently relies on detecting anomalies based on sensor behavior. The system incorporates LSTM (Long Short-Term Memory) models to predict future temperature fluctuations based on historical current data. The hypothesis is that temperature and current readings are strongly correlated, and any drift in this correlation will be flagged as anomalous, potentially signaling an impending fault.

The correlation analysis showed strong relationships between ambient temperature and the temperature sensors, suggesting that environmental factors play a significant role in asset thermal behaviour. The results indicate that the system is efficient in detecting thermal issues, although further fine-tuning is required to reduce false positives.

Overall, the system is effective for thermal monitoring and anomaly detection. Incorporated LSTM for accurate temperature predictions, reducing false positives, and refining the correlation between temperature and current to improve the detection of anomalous behavior.

APPENDICES

APPENDIX-1: CODE COMPILER

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib. pyplot as plt

data = pd.read_csv('4.1.SB-2 OUTGOING-1 UPSTREAM.csv')


data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y %H:%M',
errors='coerce')

columns_of_interest = ['S1', 'S2', 'S3', 'AT', 'CL1', 'CL2', 'CL3']

correlation_matrix = data[columns_of_interest].corr()

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

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',
square=True)

plt.title('Correlation Matrix')

#plt.show()


specific_dates = [pd.to_datetime('2024-06-10'), pd.to_datetime('2024-06-18'),
pd.to_datetime('2024-06-21'), pd.to_datetime('2024-07-08')]


# Create subplots

fig, axs = plt.subplots(3, 2, figsize=(14, 12))


# Plot for S1

axs[0, 0].plot(data['Date'], data['S1'], label='S1')

axs[0, 0].plot(data['Date'], data['AT'], label='Ambient Temperature', alpha=0.5)

for date in specific_dates:

    axs[0, 0].axvline(x=date, color='r', linestyle='--', label=str(date.date()))
```

```
axs[0, 0].set_title('S1 vs Ambient Temperature')
axs[0, 0].legend()

axs[0, 1].plot(data['Date'], data['S1'], label='S1')
axs[0, 1].plot(data['Date'], data[['CL1', 'CL2', 'CL3']], label='Current Readings', alpha=0.5)
for date in specific_dates:
    axs[0, 1].axvline(x=date, color='r', linestyle='--', label=str(date.date()))
axs[0, 1].set_title('S1 vs Current Readings')
axs[0, 1].legend()

# Plot for S2
axs[1, 0].plot(data['Date'], data['S2'], label='S2')
axs[1, 0].plot(data['Date'], data['AT'], label='Ambient Temperature', alpha=0.5)
for date in specific_dates:
    axs[1, 0].axvline(x=date, color='r', linestyle='--', label=str(date.date()))
axs[1, 0].set_title('S2 vs Ambient Temperature')
axs[1, 0].legend()

axs[1, 1].plot(data['Date'], data['S2'], label='S2')
axs[1, 1].plot(data['Date'], data[['CL1', 'CL2', 'CL3']], label='Current Readings', alpha=0.5)
for date in specific_dates:
    axs[1, 1].axvline(x=date, color='r', linestyle='--', label=str(date.date()))
axs[1, 1].set_title('S2 vs Current Readings')
axs[1, 1].legend()

# Plot for S3
axs[2, 0].plot(data['Date'], data['S3'], label='S3')
axs[2, 0].plot(data['Date'], data['AT'], label='Ambient Temperature', alpha=0.5)
```

```
for date in specific_dates:
    axs[2, 0].axvline(x=date, color='r', linestyle='--', label=str(date.date()))
axs[2, 0].set_title('S3 vs Ambient Temperature')
axs[2, 0].legend()

axs[2, 1].plot(data['Date'], data['S3'], label='S3')
axs[2, 1].plot(data['Date'], data[['CL1', 'CL2', 'CL3']], label='Current Readings', alpha=0.5)
for date in specific_dates:
    axs[2, 1].axvline(x=date, color='r', linestyle='--', label=str(date.date()))
axs[2, 1].set_title('S3 vs Current Readings')
axs[2, 1].legend()

plt.tight_layout()
plt.show()

import pandas as pd
import numpy as np
import plotly.graph_objects as go

data = pd.read_csv('4.1.SB-2 OUTGOING-1 UPSTREAM.csv')

data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y %H:%M', dayfirst=True)
data.set_index('Date', inplace=True)

sensor_columns = ['S1', 'S2', 'S3', 'CL1', 'CL2', 'CL3', 'AT']
window_size = 24
threshold = 0.8
highlight_threshold = 0.4
```

```
highlight_dates = ['2024-06-10', '2024-06-18', '2024-06-21', '2024-07-08']

for i in range(len(sensor_columns)):
    for j in range(i + 1, len(sensor_columns)):
        col1 = sensor_columns[i]
        col2 = sensor_columns[j]

        rolling_corr = data[col1].rolling(window=window_size).corr(data[col2])

        fig = go.Figure()

        fig.add_trace(go.Scatter(x=rolling_corr.index, y=rolling_corr, mode='lines',
name=f'Correlation {col1} vs {col2}', line=dict(color='blue')))

        fig.add_trace(go.Scatter(x=rolling_corr.index, y=[threshold]*len(rolling_corr),
mode='lines', name='Drift Threshold', line=dict(color='red', dash='dash')))

        drifting = rolling_corr < threshold
        fig.add_trace(go.Scatter(x=rolling_corr.index[drifting],
y=rolling_corr[drifting], mode='lines', fill='tozeroy', fillcolor='rgba(255,165,0,0.3)',
name='Drifting Periods'))

        highlight_points = rolling_corr < highlight_threshold
        fig.add_trace(go.Scatter(x=rolling_corr.index[highlight_points],
y=rolling_corr[highlight_points], mode='markers', name='Exceeds 0.4',
marker=dict(color='red', size=8)))

        for date in highlight_dates:
```

```
        fig.add_shape(type="line", x0=date, x1=date, y0=rolling_corr.min(), y1=rolling_corr.max(),
                      line=dict(color='green', dash='dash', width=2), name='Highlight Date')

    fig.update_layout(title=f'Rolling Correlation Between {col1} and {col2}',
                      xaxis_title='Date',
                      yaxis_title='Correlation Coefficient',
                      xaxis_tickformat="%Y-%m-%d",
                      legend=dict(x=0, y=1),
                      template='plotly_white')

    fig.show()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


data = pd.read_csv("4.1.SB-2 OUTGOING-1 UPSTREAM.csv")

sensor_columns = ['S1', 'S2', 'S3', 'CL1', 'CL2', 'CL3', 'AT']

data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y %H:%M', dayfirst=True)
data.set_index('Date', inplace=True)

samples_per_hour = 4
expected_samples_per_day = samples_per_hour * 24
```

```
data_loss = data[sensor_columns].isnull().sum(axis=1)

daily_data_loss = data_loss.resample('D').sum()

threshold = expected_samples_per_day * 0.1  # For example, 10% loss is considered high
daily_data_loss_flagged = daily_data_loss[daily_data_loss > threshold]

plt.figure(figsize=(12, 6))
plt.plot(daily_data_loss.index, daily_data_loss, label='Daily Data Loss', color='blue')

plt.scatter(daily_data_loss_flagged.index, daily_data_loss_flagged, color='red', label='High Data Loss', zorder=5)

plt.title('Daily Frequency of Data Loss', fontsize=16)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Number of Missing Samples', fontsize=12)
plt.axhline(y=threshold, color='orange', linestyle='--', label='Threshold (10% of expected samples)')
plt.legend()
plt.grid()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
import pandas as pd

import numpy as np

import plotly.graph_objects as go


data = pd.read_csv('4.1.SB-2 OUTGOING-1 UPSTREAM.csv')


data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y %H:%M',
dayfirst=True)

data.set_index('Date', inplace=True)


sensor_columns = ['S1', 'S2', 'S3', 'CL1', 'CL2', 'CL3', 'AT']

window_size = 24

threshold = 0.8

highlight_threshold = 0.4

highlight_dates = ['2024-06-10', '2024-06-18', '2024-06-21', '2024-07-08']


for i in range(len(sensor_columns)):

    for j in range(i + 1, len(sensor_columns)):

        col1 = sensor_columns[i]

        col2 = sensor_columns[j]


        rolling_corr = data[col1].rolling(window=window_size).corr(data[col2])
```

```
fig = go.Figure()

fig.add_trace(go.Scatter(x=rolling_corr.index, y=rolling_corr, mode='lines',
name=f'Correlation {col1} vs {col2}', line=dict(color='blue')))

fig.add_trace(go.Scatter(x=rolling_corr.index, y=[threshold]*len(rolling_corr),
mode='lines', name='Drift Threshold', line=dict(color='red', dash='dash')))

drifting = rolling_corr < threshold

fig.add_trace(go.Scatter(x=rolling_corr.index[drifting],
y=rolling_corr[drifting], mode='lines', fill='tozeroy', fillcolor='rgba(255,165,0,0.3)',
name='Drifting Periods'))

highlight_points = rolling_corr < highlight_threshold

fig.add_trace(go.Scatter(x=rolling_corr.index[highlight_points],
y=rolling_corr[highlight_points], mode='markers', name='Exceeds 0.4',
marker=dict(color='red', size=8)))

for date in highlight_dates:

    fig.add_shape(type="line", x0=date, x1=date, y0=rolling_corr.min(),
y1=rolling_corr.max(),
```

```
                line=dict(color='green', dash='dash', width=2), name='Highlight
Date')

    fig.update_layout(title=f'Rolling Correlation Between {col1} and {col2}',
                xaxis_title='Date',
                yaxis_title='Correlation Coefficient',
                xaxis_tickformat="%Y-%m-%d",
                legend=dict(x=0, y=1),
                template='plotly_white')

    fig.show()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv('4.1.SB-2 OUTGOING-1 UPSTREAM.csv')

data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y %H:%M',
dayfirst=True)
data.set_index('Date', inplace=True)

sensor_columns = ['S1', 'S2', 'S3', 'CL1', 'CL2', 'CL3', 'AT']
window_size = 20  # 5 hours (4 samples per hour * 5 hours)
```

```
threshold = 0.8
highlight_threshold = 0.4

drift_flags = {}
highlight_dates = ['2024-06-10', '2024-06-18', '2024-06-21', '2024-07-08']

for i in range(len(sensor_columns)):
    for j in range(i + 1, len(sensor_columns)):
        col1 = sensor_columns[i]
        col2 = sensor_columns[j]

        rolling_corr = data[col1].rolling(window=window_size).corr(data[col2])

        drifting = rolling_corr < threshold
        drift_flags[(col1, col2)] = drifting

        plt.figure(figsize=(12, 6))
        plt.plot(rolling_corr, label=f'Correlation {col1} vs {col2} (Window: 5 hours)',
color='blue')
        plt.axhline(y=threshold, color='red', linestyle='--', label='Drift Threshold')
        plt.fill_between(rolling_corr.index, rolling_corr, threshold, where=drifting,
                         color='orange', alpha=0.3, label='Drifting Periods')
```

```
highlight_points = rolling_corr < highlight_threshold
plt.scatter(rolling_corr.index[highlight_points], rolling_corr[highlight_points], color='red', label='Exceeds 0.4', zorder=5)

for date in highlight_dates:
    plt.axvline(pd.to_datetime(date), color='black', linestyle='--', label='Highlight Date', alpha=0.7)

plt.title(f'Rolling Correlation Between {col1} and {col2}', fontsize=16)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Correlation Coefficient', fontsize=12)
plt.legend()
plt.grid()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
import pandas as pd

import numpy as np

import plotly.graph_objects as go


data = pd.read_csv('4.1.SB-2 OUTGOING-1 UPSTREAM.csv')


data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y %H:%M',
dayfirst=True)

data.set_index('Date', inplace=True)


sensor_columns = ['S1', 'S2', 'S3', 'CL1', 'CL2', 'CL3', 'AT']


window_size = 24

highlight_threshold = 0.4


highlight_dates = ['2024-06-10', '2024-06-18', '2024-06-21', '2024-07-08']


for col in sensor_columns:


    rolling_std = data[col].rolling(window=window_size).std()


    fig = go.Figure()
```

```
    fig.add_trace(go.Scatter(x=rolling_std.index, y=rolling_std, mode='lines',
name=f'Rolling Std Dev of {col}', line=dict(color='blue')))

    fig.add_trace(go.Scatter(x=rolling_std.index,
y=[highlight_threshold]*len(rolling_std), mode='lines', name=f'Threshold:
{highlight_threshold}', line=dict(color='red', dash='dash')))

    deviation_exceeds = rolling_std > highlight_threshold
    fig.add_trace(go.Scatter(x=rolling_std.index[deviation_exceeds],
y=rolling_std[deviation_exceeds],
                             mode='lines', fill='tozeroy', fillcolor='rgba(255,165,0,0.3)',
                             name=f'Deviation > {highlight_threshold}'))

    for date in highlight_dates:
        fig.add_shape(type="line", x0=date, x1=date, y0=rolling_std.min(),
y1=rolling_std.max(),
                      line=dict(color='green', dash='dash', width=2), name='Highlight Date')

    fig.update_layout(title=f'Rolling Standard Deviation for {col}',
                      xaxis_title='Date',
                      yaxis_title='Standard Deviation',
                      xaxis_tickformat="%Y-%m-%d",
                      template='plotly_white',
                      legend=dict(x=0, y=1))
```

```
    fig.show()


import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.stats import zscore


data = pd.read_csv("4.1.SB-2 OUTGOING-1 UPSTREAM.csv")


data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y %H:%M',
dayfirst=True)

data.set_index('Date', inplace=True)


sensor_columns = ['S1', 'S2', 'S3', 'CL1', 'CL2', 'CL3', 'AT']


plt.figure(figsize=(12, 8))


for col in sensor_columns:


    data[f'{col}_zscore'] = zscore(data[col].dropna())


    outliers = data[np.abs(data[f'{col}_zscore']) > 3]
```

```
    plt.plot(data.index, data[col], label=f'{col} Data', alpha=0.5)

    plt.scatter(outliers.index, outliers[col], color='red', label=f'{col} Outliers',
zorder=5)

event_dates = pd.to_datetime(['2024-06-10', '2024-06-18', '2024-06-21',
'2024-07-08'])

for event in event_dates:
    plt.axvline(x=event, color='black', linestyle='--', label='Event Date' if event ==
event_dates[0] else "")

plt.title('Outlier Detection Across Multiple Sensors Using Z-Score', fontsize=16)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Sensor Values', fontsize=12)
plt.legend()
plt.grid()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
import pandas as pd
import matplotlib.pyplot as plt

datasets = [
    pd.read_csv('Transformer Room.csv'),
    pd.read_csv('Transformer Room(SIPCOT).csv'),
    pd.read_csv('Transformer ACB Room.csv'),
    pd.read_csv('Tower B SWG Room.csv')
]



vertical_lines = [
    ['2024-04-06'],
    ['2024-06-04'],
    ['2024-04-09'],
    ['2024-04-04','2024-06-13']
]



fig, axs = plt.subplots(2, 2, figsize=(12, 8))
axs = axs.flatten()
```

```
titles = [
    'Transformer Room',
    'Transformer Room (SIPCOT)',
    'Transformer ACB Room',
    'Tower B SWG Room'
]

for i, data in enumerate(datasets):
    # Specify the date format
    data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y %H:%M',
dayfirst=True)
    axs[i].plot(data['Date'], data['AT'], label=f'Dataset {i + 1}')
    axs[i].set_title(titles[i])
    axs[i].set_xlabel('Date (Month Year)')
    axs[i].set_ylabel('Ambient Temperature (AT)')
    axs[i].legend()



    axs[i].xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%b %Y'))
    axs[i].tick_params(axis='x', rotation=45)
    axs[i].set_xticks(data['Date'][::len(data) // 6])
```

```
    for vline_date in vertical_lines[i]:
        axs[i].axvline(pd.to_datetime(vline_date), color='red', linestyle='--',
                       label='Event Date' if vline_date == vertical_lines[i][0] else "")


handles, labels = axs[-1].get_legend_handles_labels()
by_label = dict(zip(labels, handles))
axs[-1].legend(by_label.values(), by_label.keys())


plt.tight_layout()
plt.show()
```

APPENDIX-2:SCREENSHOTS

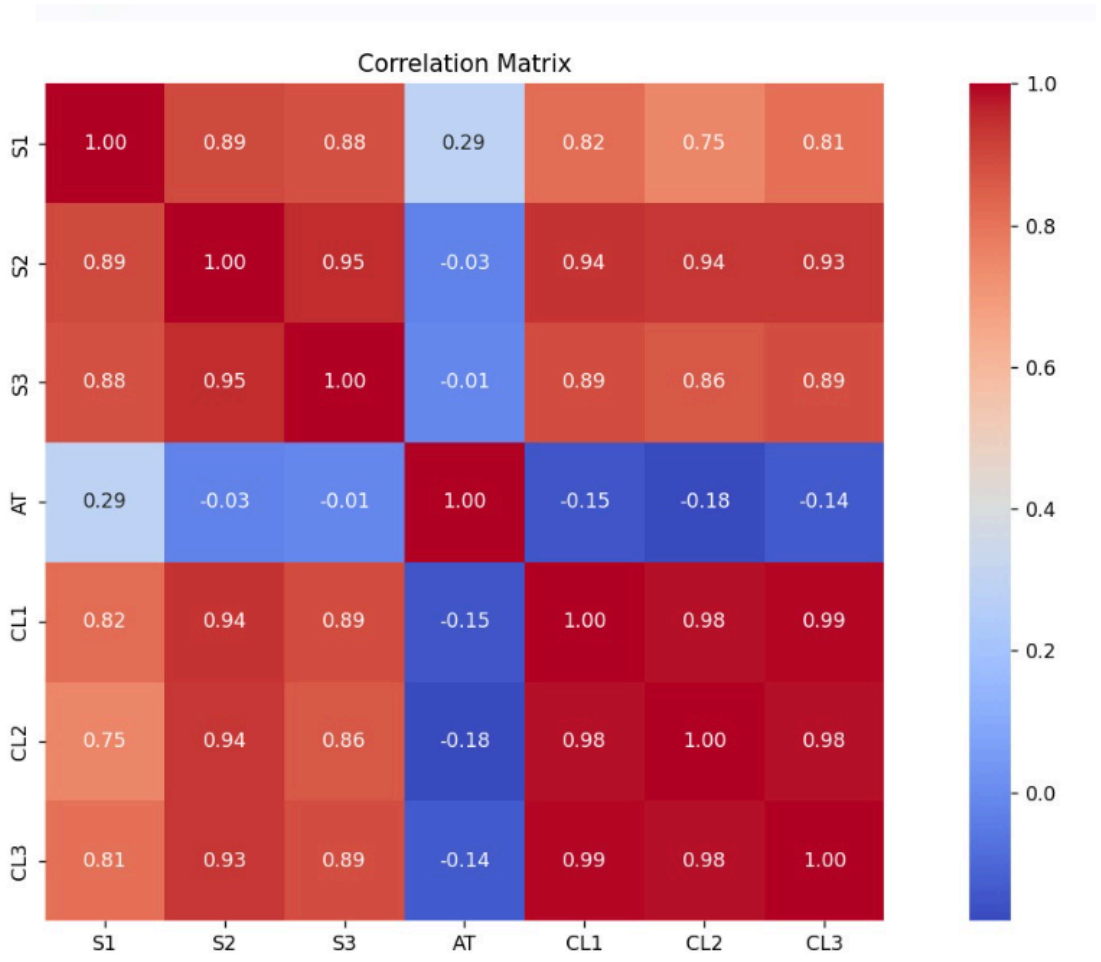

fig.1

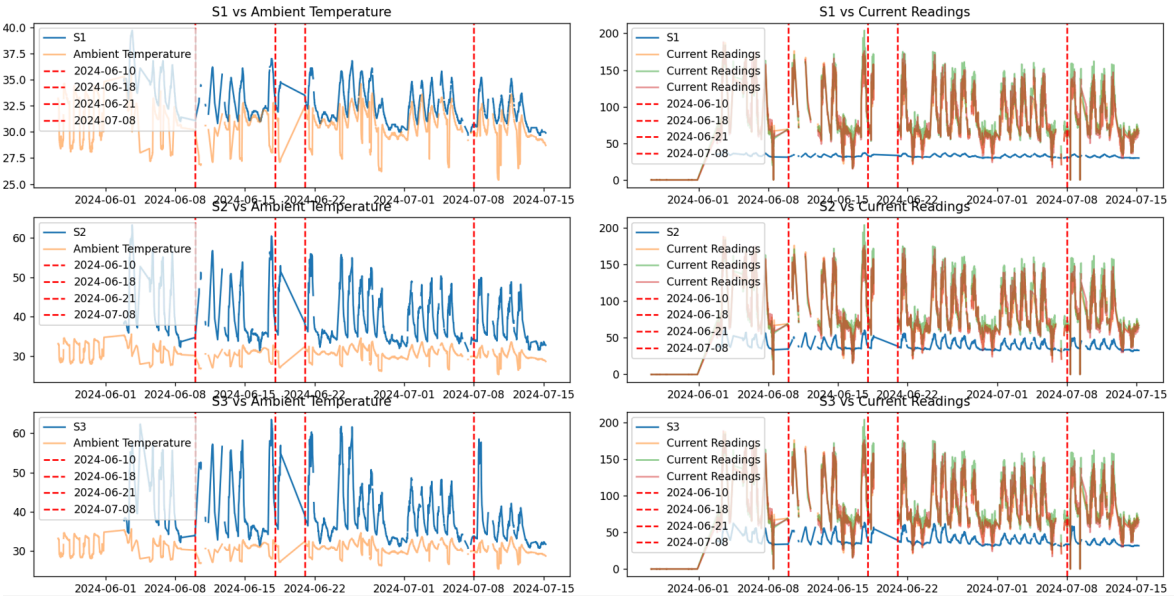

fig.2

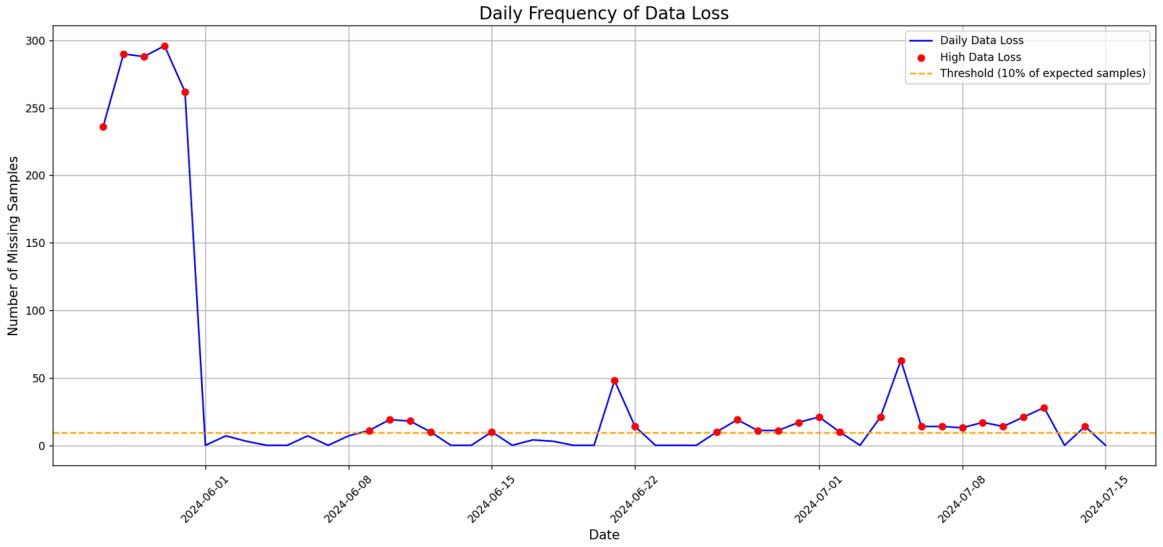

fig.3

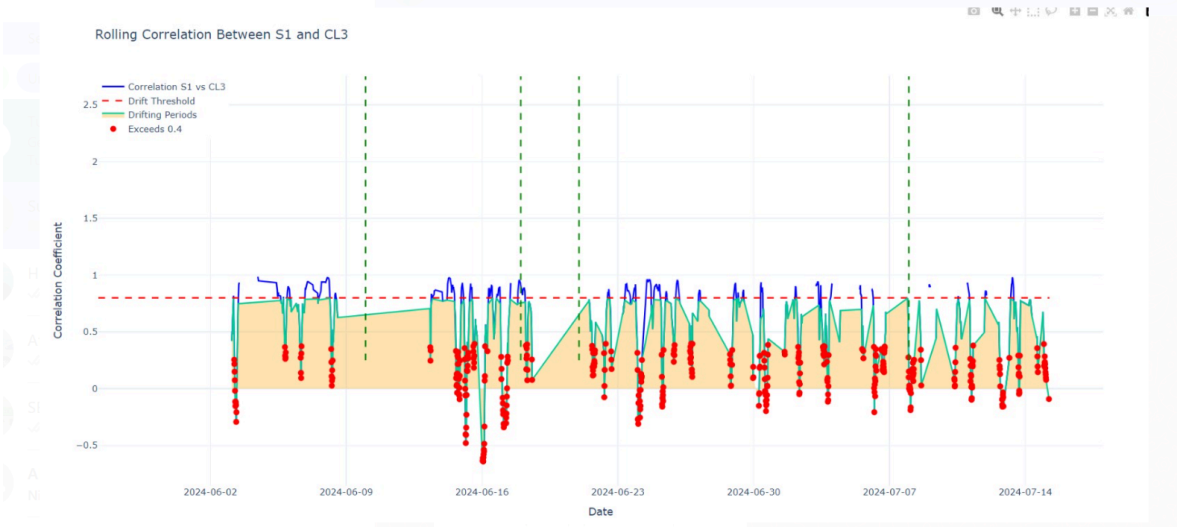

fig.4

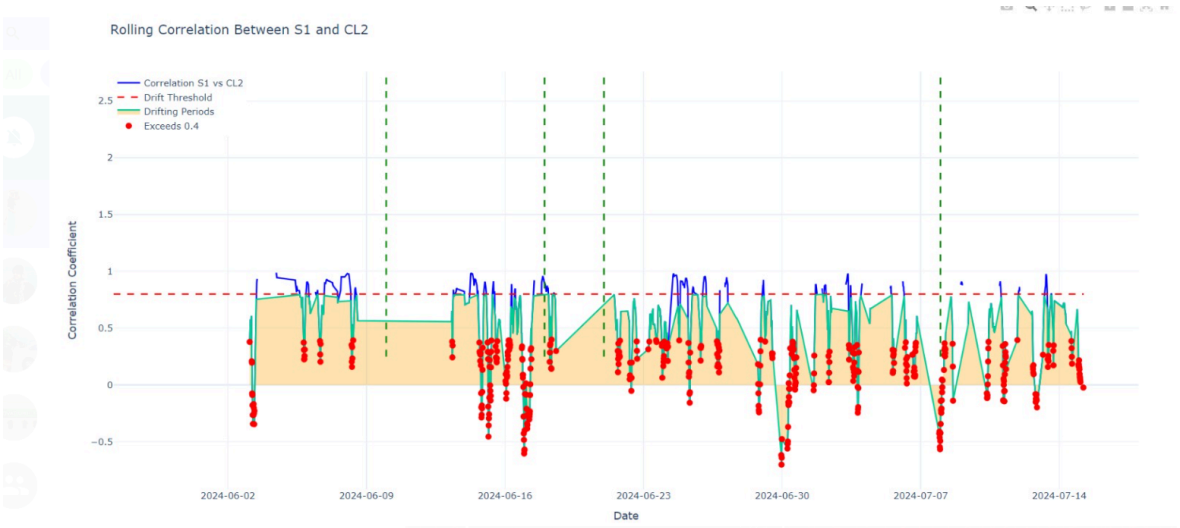

fig.5

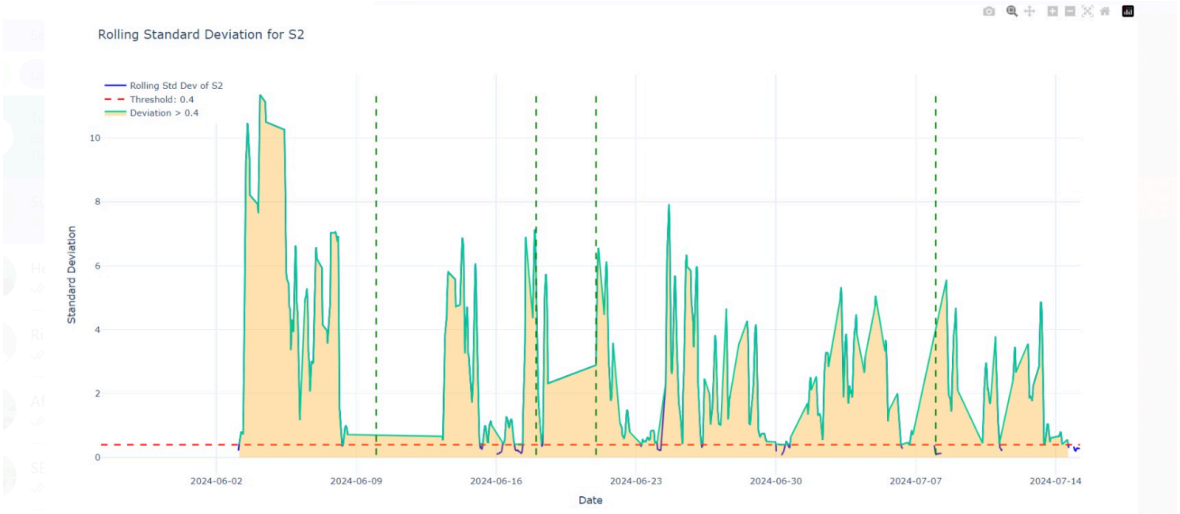

fig.6

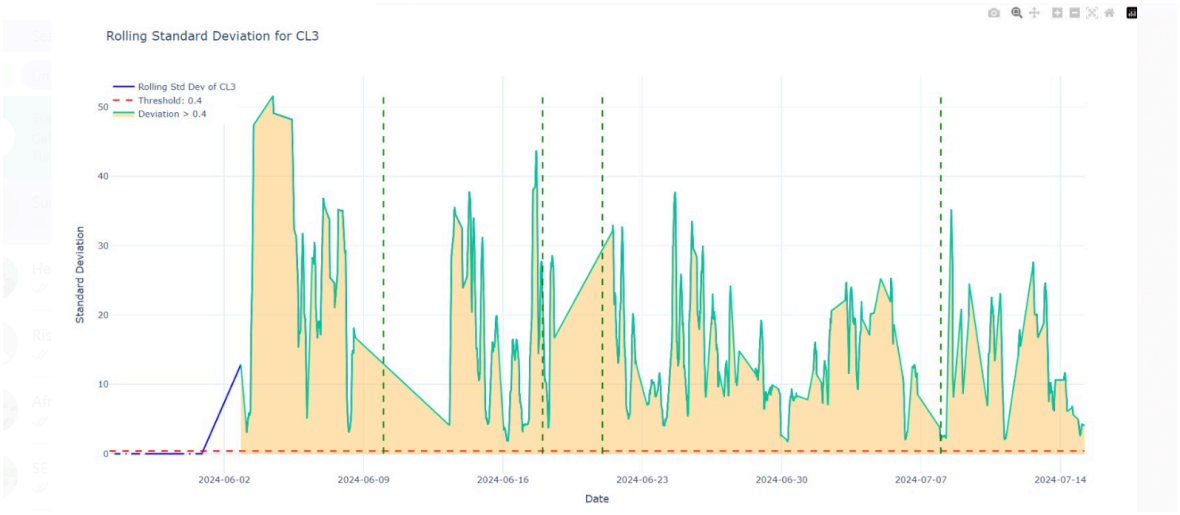
Rolling Standard Deviation for CL3
Rolling Std Dev of CL3
Threshold: 0.4
Deviation > 0.4
Standard Deviation
0
10
20
30
40
50
2024-06-02
2024-06-09
2024-06-16
2024-06-23
2024-06-30
2024-07-07
2024-07-14
Date

fig.7

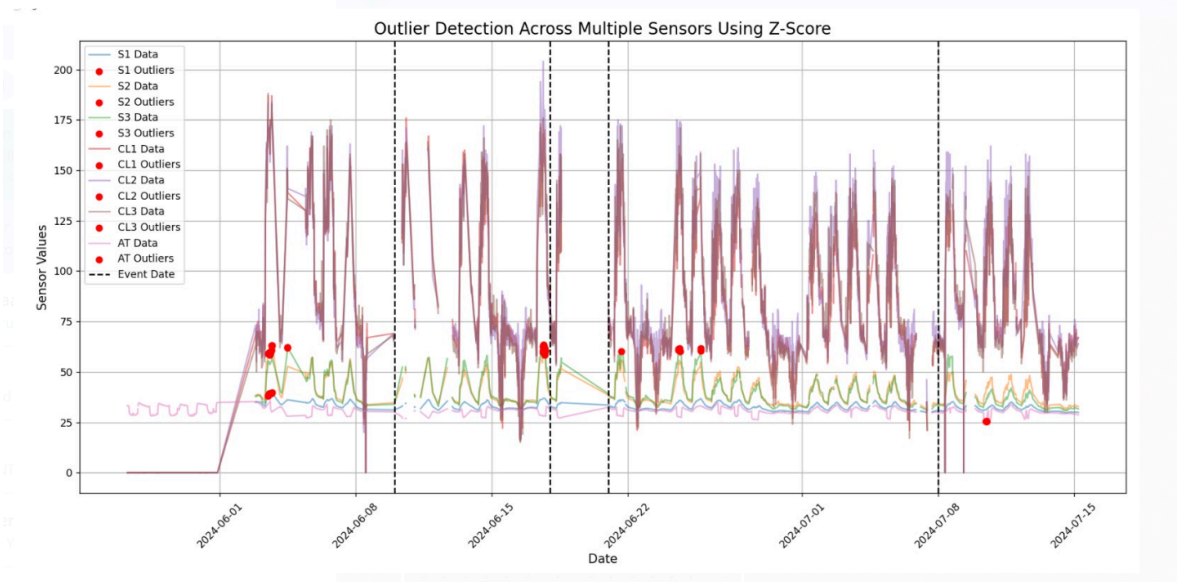
Outlier Detection Across Multiple Sensors Using Z-Score
S1 Data
S1 Outliers
S2 Data
S2 Outliers
S3 Data
S3 Outliers
CL1 Data
CL1 Outliers
CL2 Data
CL2 Outliers
CL3 Data
CL3 Outliers
AT Data
AT Outliers
Event Date
Sensor Values
0
25
50
75
100
125
150
175
200
2024-06-01
2024-06-08
2024-06-15
2024-06-22
2024-07-01
2024-07-08
2024-07-15
Date

fig.8

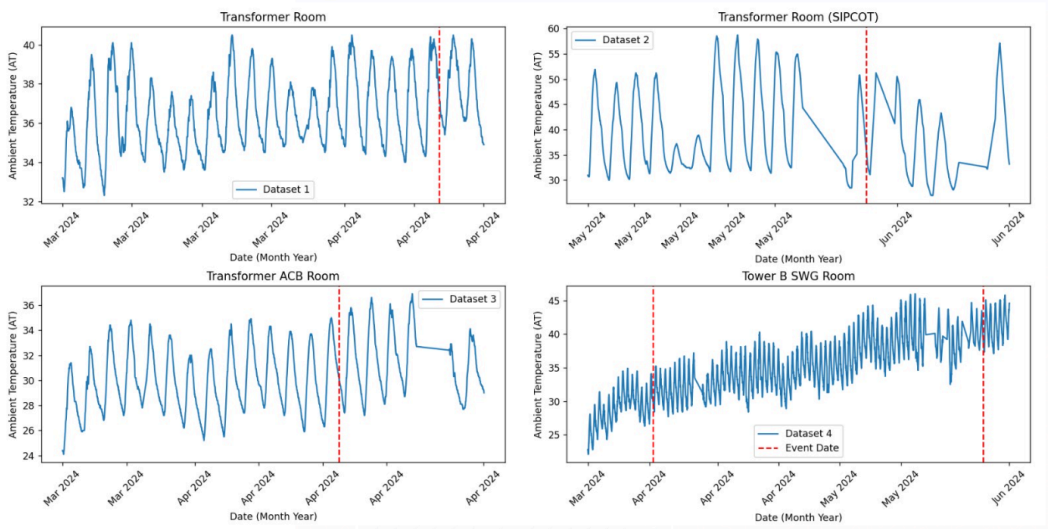
Transformer Room
Ambient Temperature (AT)
Dataset 1
Date (Month Year)
Transformer Room (SIPCOT)
Dataset 2
Transformer ACB Room
Dataset 3
Tower B SWG Room
Dataset 4
Event Date

fig.9

REFERENCES

1. *Li, X., Zhang, Y., & Wang, J. (2021). Infrared thermography in industrial applications. *Infrared Physics and Technology, 113, 103536. doi:10.1016/j.infrared.2021.103536.

2. *Smith, J., & Johnson, R. (2020). Advancements in thermocouple technology. *Engineering Innovations, 27(3), 189-202. doi:10.1016/j.enginnov.2020.05.003.

3. *Jones, A., & Green, L. (2019). The role of thermistors in medical applications. *Healthcare Technology Review, 14(2), 67-75. doi:10.1016/j.HTR.2019.01.006.

4. *Patel, R., Kumar, V., & Singh, A. (2022). IoT in thermal monitoring systems. *Journal of Internet of Things, 8(4), 122-134. doi:10.1016/j.iot.2022.101241.

5. *Nguyen, T., & Tran, H. (2023). Machine learning for predictive thermal analytics. *International Journal of Data Science, 15(1), 45-60. doi:10.1016/j.ijdatasci.2023.01.002.

6. *Kumar, R., Singh, P., & Verma, S. (2020). Thermal monitoring in industrial processes. *Journal of Industrial Technology, 36(2), 14-22. doi:10.1080/15236934.2020.1740896.

7. *Wang, Y., Zhao, X., & Liu, T. (2021). Smart building temperature control systems. *Building Science Journal, 56(1), 81-93. doi:10.1016/j.buildsci.2021.102034.

8. *Lee, C., Park, S., & Kim, H. (2022). Remote patient monitoring systems: A review. *Journal of Telemedicine, 28(3), 245-256. doi:10.1016/j.jtele.2022.02.005.

9. *Brown, T., & White, J. (2021). Calibration techniques for thermal sensors. *Journal of Measurement Science, 35(4), 327-339. doi:10.1016/j.jms.2021.04.005.

10. *Zhang, L., Li, P., & Wang, F. (2023). Cybersecurity challenges in IoT thermal monitoring. *Cybersecurity Journal, 9(2), 112-126. doi:10.1016/j.cyber.2023.06.003.

11. *Kim, S., Lee, J., & Chen, R. (2024). Nanomaterials for thermal sensors: Future trends. *Materials Science Advances, 19(1), 25-38. doi:10.1016/j.msa.2023.02.007.

WORKLOG

Date	Task Done
26/08/2024-28/08/2024	Onboarding
29/08/2024	Laptop VPN configuration
30/08/2024	Asset Advisor Team Meeting
02/09/2024-04/09/2024	Ecostruxure power KT
05/09/2024	Admin and access rights
06/09/2024	Pycharm and Python Demo
09/09/2024-10/09/2024	Data structure and python demo
11/09/2024-12/09/2024	File manipulation in pydemo
13/09/2024-17/09/2024	Customer csv file analysis
18/03/2024-25/09/2024	Data identification and collection
26/09/2024-03/10/2024	Data preprocessing
07/10/2024-11/10/2024	Data Visualisation and analysis
12/10/2024-14/10/2024	Experimenting with algorithms
15/10/2024-17/10/2024	Model Development