**Indoor Air Quality Monitoring with Arduino IoT Devices**

*Marquez, Gabriel Edward P. |* [*60300343@udst.edu.qa*](mailto:60300343@udst.edu.qa)

*Hanif, Ahmed |* [*60301085@udst.edu.qa*](mailto:60301085@udst.edu.qa)

**Introduction**

Indoor air quality (IAQ) is directly linked to occupant health, comfort, and productivity in confined spaces like homes, schools, and offices. IAQ can be affected by various chemicals, including gases (i.e., carbon monoxide, ozone, radon), volatile organic compounds (VOCs), particulate matter (PM) and fibers, organic and inorganic contaminants, and biological particles such as bacteria, fungi, and pollen [1]. Indoor air is a dominant exposure for humans, but has been a rather late consideration against earlier environmental issues such as energy use, sustainability and outdoor air quality [2].

In this report, an IoT system that involves an Arduino MKR1010 board and built-in temperature and CO₂ sensors is employed to track temperature and CO₂ levels in the span of 10 days. The data is collected via the Arduino Cloud API implemented in Jupyter notebook, and is fed to a Python-based Seasonal Autoregressive Integrated Moving Average (SARIMA) model to perform time series analysis on the sensor data collected.

This report aims to continuously track CO₂ and temperature values for the development of smart IAQ monitoring systems which enable timely ventilation adjustments for healthier indoor environments.

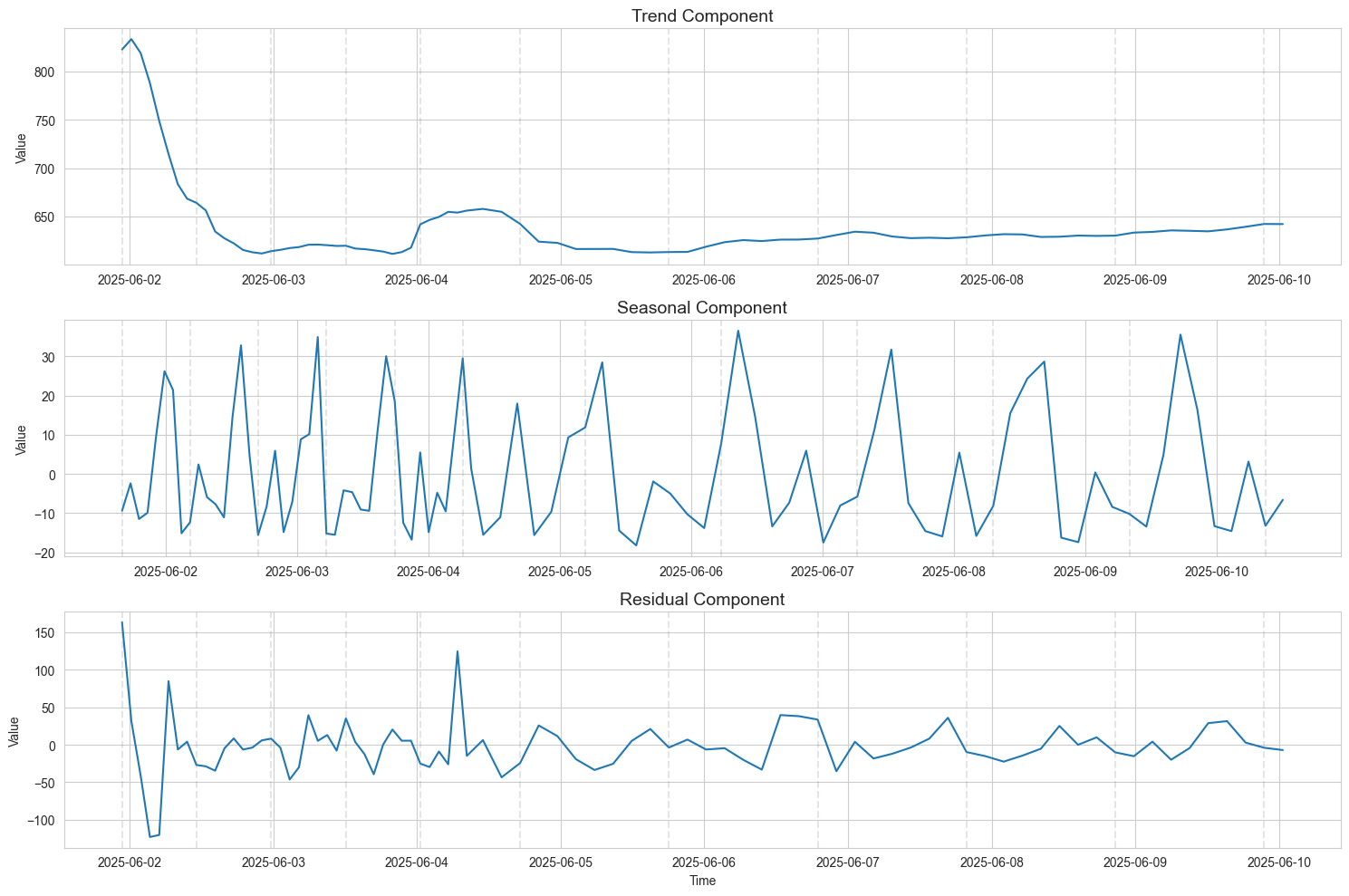
**Methodology**

This project involves two parts: gathering temperature and CO₂ data through the Arduino Cloud platform, and implementing the SARIMA model with the gathered sensor data.

The data gathering is initiated by connecting an Arduino MKR1010 board to a PC. This involves the configuration of the device to the Arduino Cloud platform, and the physical connection required for the device and the Arduino service. This device is connected to a Thing created in the Arduino Cloud platform inside the school organization. A dashboard is also generated to track real-time sensor readings, alongside providing historical data inputted by the sensors. The Arduino API documentation has been the basis of this project to gain access to the IoT Cloud data. The information needed to gather the data such as the organization ID, and an API key alongside a given secret has been generated and taken through the Arduino IoT Cloud platform.

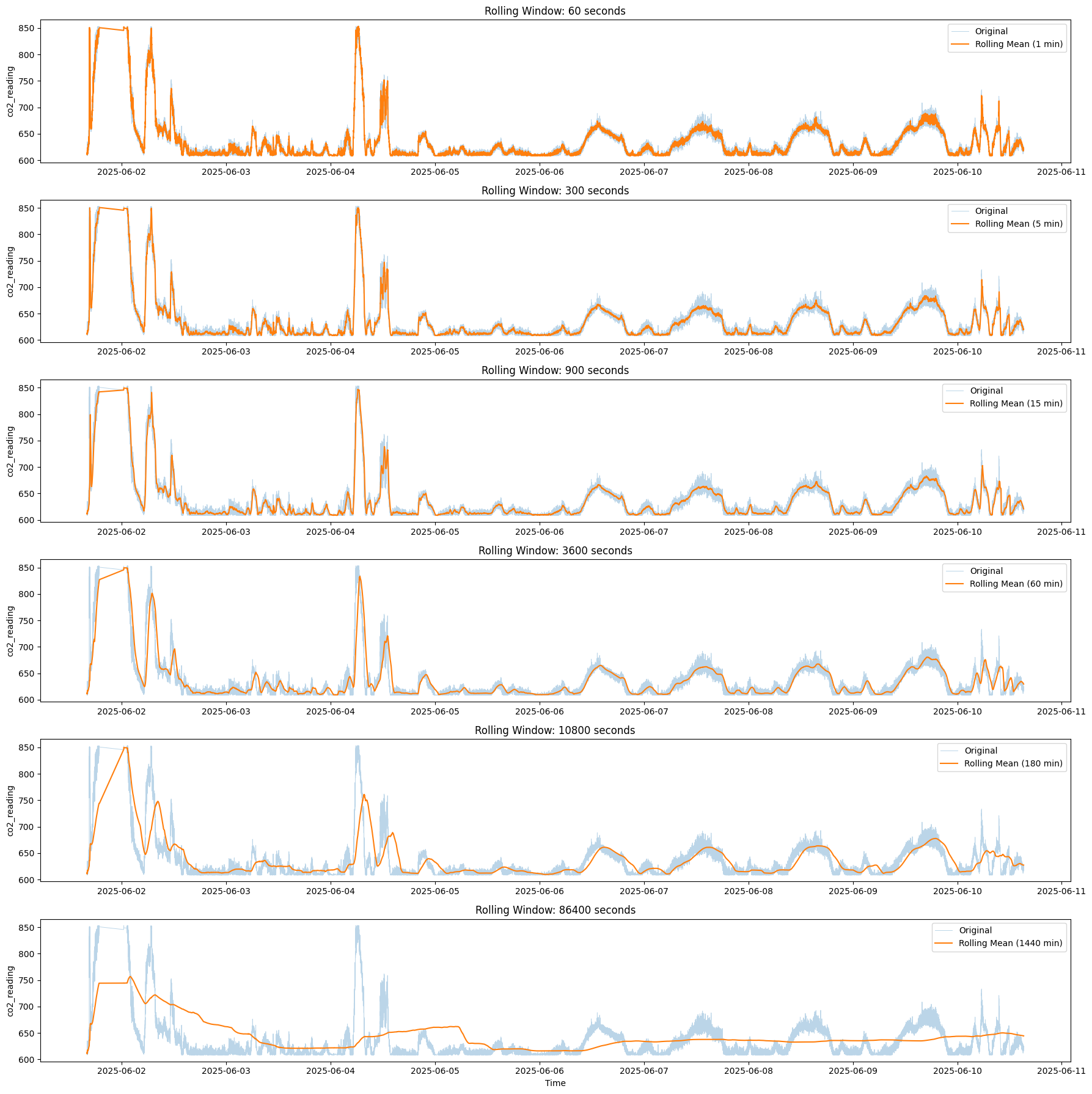
To fetch the data generated by the Arduino board, a Python application implemented in a Jupyter notebook has been developed to retrieve telemetry data from the Arduino IoT Cloud. In order to take specific data within the organization, the organization ID, API key, and API secret have been used to generate a client. Once a client token is generated, two API clients are initialized: namely, the PropertiesV2Api, for accessing metadata about the device’s properties, and SeriesV2Api, for retrieving time-series telemetry data. To provide continuous fetching of data, a workaround that involves a while loop is used to bypass timeouts implemented by the API. Finally, the data is stored in a CSV file, which is constantly appended as the application fetches new data.

**Exploratory Data Analysis**



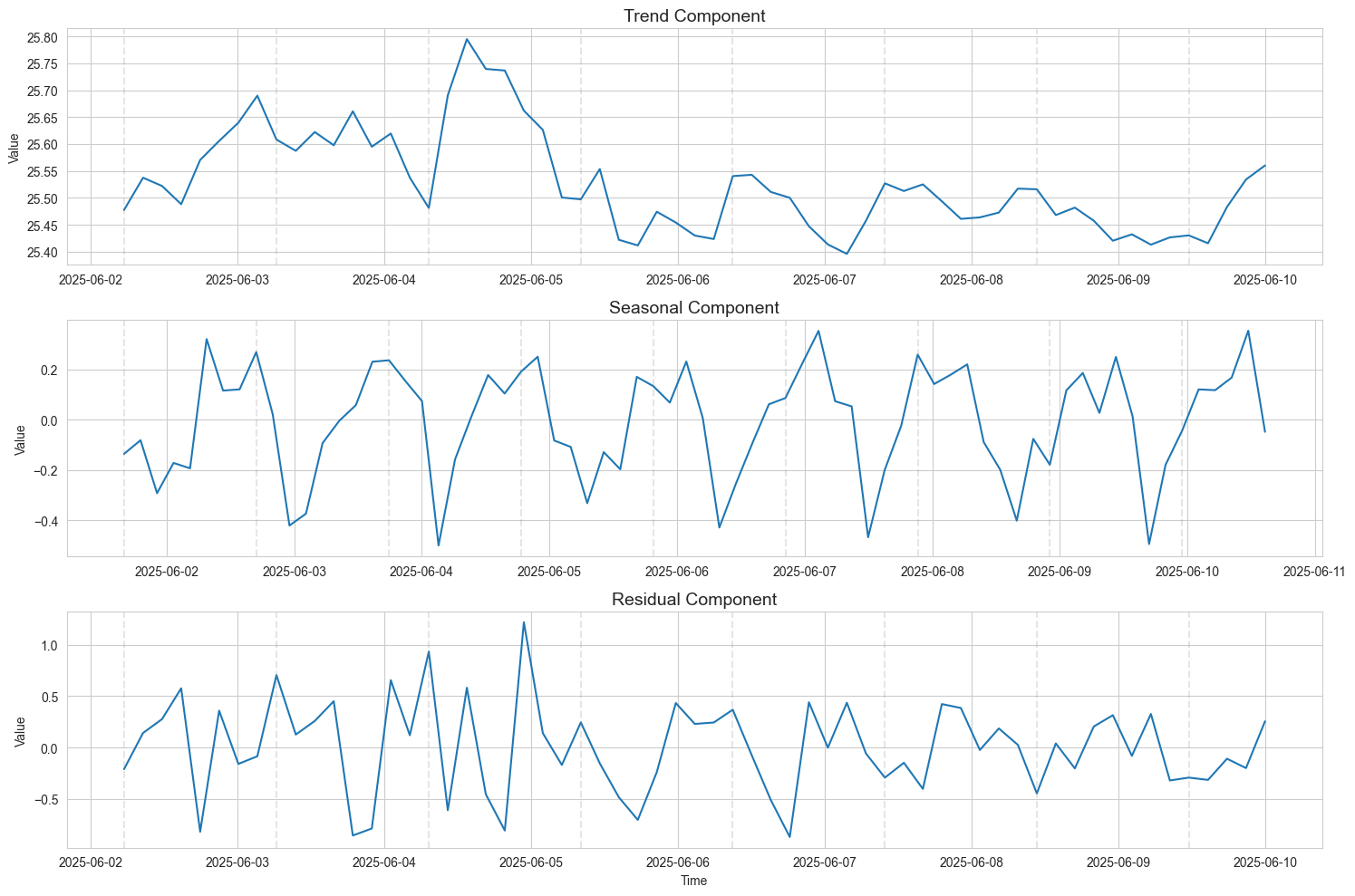
*Figure 1. CO₂ Sensor Data Trends, Seasonality, Residuals*

CO₂ concentration peaked above 800 ppm during June 2, following a sharp decline after. Moreover, seasonality is evident due to the repeating daily fluctuations, which likely simulates people occupying the room during working hours. In addition, irregular activity is evident during June 2, which may be related to the anomaly found in the trend component. From June 4 onwards, residuals have become more minimal and have been deemed more stable, indicating an improved environmental control or sensor calibration, alongside consistent seasonality.



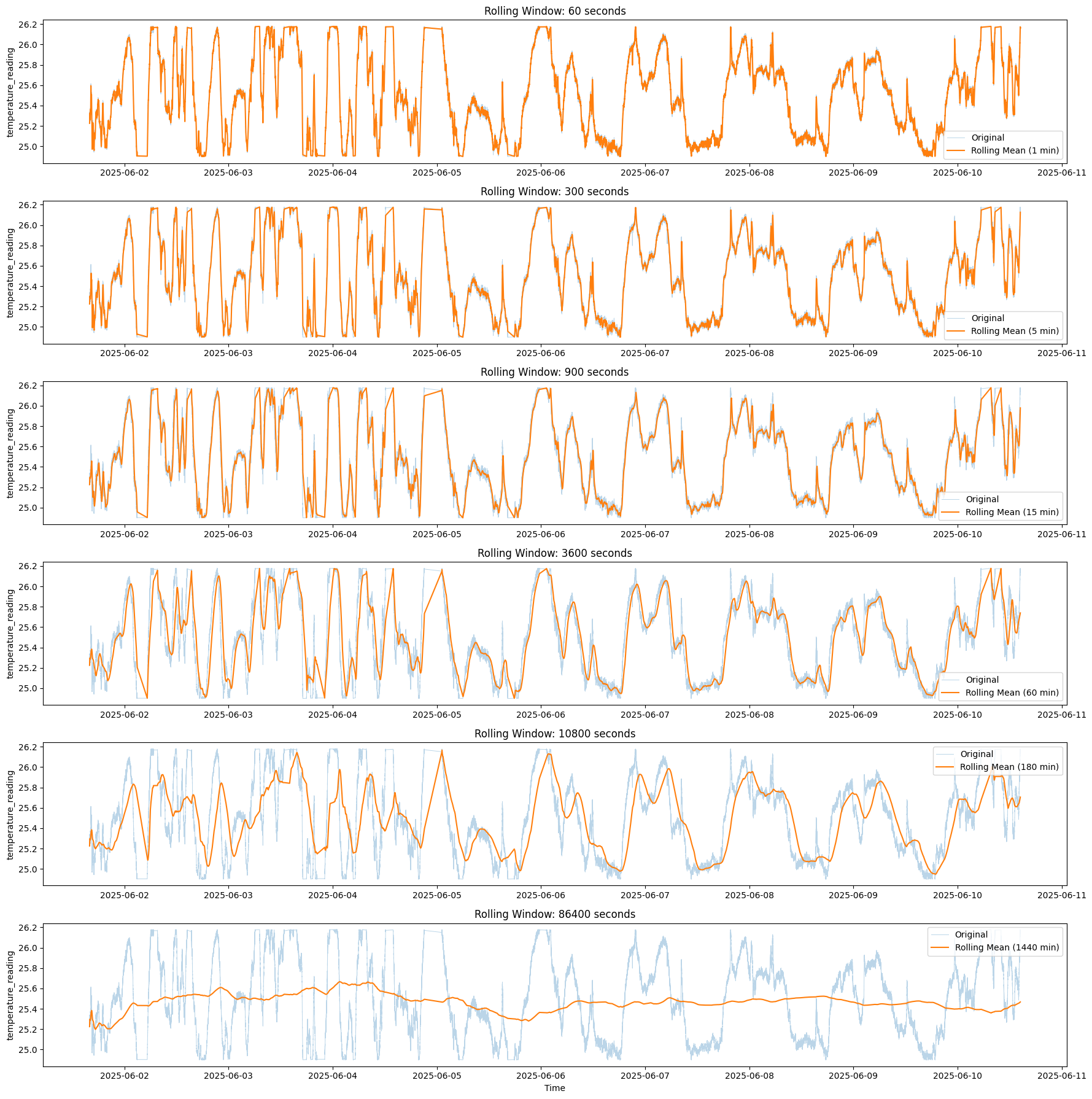
*Figure 2. CO₂ Concentration Rolling Window Comparisons*

This figure shows the original CO₂ data with rolling averages using various time windows: every one minute, every 5 minutes, every 15 minutes, every 1 hour, every 3 hours, and every day. Taking the data after 1-15 minutes capture short-term fluctuations, which may be ideal for real-time alerting but not helpful for our current implementation. On the other hand, taking it every 3 hours-1 day gives a very smooth daily trend, which dilutes the real pattern of the CO₂ data gathered. The most ideal time window is every hour, as it is smoother while identifying daytime activity cycles. Taking data every hour makes it more informative for facility management, which is very important in the context of indoor air quality monitoring.



*Figure 3. Temperature Sensor Data Trends, Seasonality, Residuals*

Temperature levels rapidly rose briefly during June 5, following a steady decline after. Moreover, seasonality is somewhat evident due to a sudden drop every start of the day. However, in contrast to the CO₂ data gathered, residuals can be seen throughout the temperature data, which can imply that the data gathered is not reliable for model implementation.



*Figure 4. Temperature Sensor Data Rolling Window Comparisons*

Similar to the CO₂ data, this figure shows the original temperature data with rolling averages using various time windows: every one minute, every 5 minutes, every 15 minutes, every 1 hour, every 3 hours, and every day. Taking the data after 1-15 minutes provided a better representation of the data generally, but is unusable for model training due to memory leaks. On the other hand, taking aggregated data per day diverges away from the real data, proving its inadequacy for the models. Thus, taking aggregated data between 1-3 hours provides better representation of the data while decreasing dataset size.

**Implementation**

A range of machine learning and statistical models have been deployed in this project to forecast CO₂ and temperature values. These models have been selected to provide diverse modeling approaches. ARIMA and SARIMAX are implemented to exhibit time series forecasting methods, a Linear Regression model as the baseline, and ensemble-based models such as XGBoost, LightGBM, Random Forest, and Gradient Boosting are used to predict the targets. The diverse selection is curated to compare the abilities of time series models to capture temporal dependencies, alongside machine learning models that provide better nonlinear predictions. Thus, the aim of this report is to evaluate the predictive performance of the following models in forecasting CO₂ and temperature data.

**Evaluation & Analysis**

This part of the report presents a comparative evaluation of various predictive models used for forecasting CO₂ and temperature levels, based on three key performance metrics: R-Squared, Mean Squared Error (MSE), and Mean Absolute Error (MAE).

| **Model** | **R-Squared** | **MSE** | **MAE** |
| --- | --- | --- | --- |
| SARIMAX | -0.7151 | NaN | 19.769644 |
| Linear Regression | -4.2039 | 45.6504 | 24.296766 |
| XGBoost | -35.5561 | 82.5764 | 48.977156 |
| Light GBM | -18.4823 | 78.8633 | 35.801374 |
| Random Forest | -35.8835 | 84.1874 | 50.661969 |
| Gradient Boosting | -36.1102 | 83.2558 | 49.207480 |
| ARIMA | -1.3696 | NaN | 23.377832 |

*Figure 5. CO₂ Model Comparisons*

Both time-series models produced undefined MSE values but relatively low MAE scores, with SARIMAX outperforming the others (MAE = 19.77). However, all models had negative R-Squared values, which indicates poor model fit. Given that, none of the models have accurately captured the variation in the data. Surprisingly, ensemble models such as XGBoost, LightGBM, Random Forest, and Gradient Boosting produced greater MSE and MAE values, with Gradient Boosting having the lowest R-Squared score (-36.1102). Despite yielding a negative R-Squared value, Linear Regression had the lowest MSE (45.65). These results prove that this dataset exhibits difficulty in estimating CO₂ levels, and further data preprocessing must be addressed to provide more meaningful predictions.

| **Model** | **R-Squared** | **MSE** | **MAE** |
| --- | --- | --- | --- |
| SARIMAX | 0.084538 | NaN | 0.236845 |
| Linear Regression | -0.016896 | 0.095546 | 0.251079 |
| XGBoost | -0.327059 | 0.124689 | 0.293803 |
| Light GBM | 0.107912 | 0.083819 | 0.237893 |
| Random Forest | -0.139951 | 0.107108 | 0.274687 |
| Gradient Boosting | -0.479987 | 0.139058 | 0.302621 |
| ARIMA | -0.087867 | NaN | 0.245911 |

*Figure 6. Temperature Model Comparisons*

Compared to the CO₂ prediction results in Figure 5, some models in this test produced somewhat positive R-Squared values, indicating a minimal capacity to explain variance in the data. Most other models, including Gradient Boosting and XGBoost, had negative R-Squared values, similar to CO₂ results. Notably, LightGBM performed the best overall, with the lowest MSE (0.0838), MAE (0.2379), and R-Squared (0.1079), indicating that it is the most effective model among those examined. SARIMAX also had a positive R-Squared (0.0845) and a competitive MAE (0.2368), but the MSE was missing. These results indicate that tree-based models, notably LightGBM, perform better for temperature forecasting in this dataset.

**Conclusion**

The evaluation results indicate that forecasting CO₂ and temperature values using both time series and machine learning models presents significant challenges, particularly in modeling CO₂ levels.

Across all models, R-Squared values for CO₂ prediction were negative, suggesting poor fit and limited ability to explain the variance in the target variable. SARIMAX, while yielding the lowest MAE for CO₂, had an undefined MSE and still showed a negative R-Squared, highlighting the difficulty of accurately modeling the data. For temperature prediction, model performance was relatively better. LightGBM outperformed other models with the lowest MSE (0.0838), low MAE (0.2379), and the highest positive R-Squared (0.1079), indicating that tree-based ensemble methods may be more suitable for capturing patterns in temperature data. SARIMAX also showed promise, achieving a competitive MAE and a slightly positive R-Squared.

Runtime issues, memory leaks, API limitations and other issues were encountered across the whole timeline of the project. During the data gathering, a workaround is implemented to continuously gather data for the project, as API limitations could only gather 1000 data points at a time. Moreover, the data gathered is fairly large, thus fitting over 1.6 million entries made encountering memory leaks more often than usual. Data modelling took more time when fitting the model for ARIMA models, making the timeline of the work less efficient. As a solution, an aggregated version of the whole dataset is used to perform time-series forecasting.

To improve model performance in future work, several steps can be taken. First and foremost, incorporating more features, such as time-based variables, weather conditions, or external environmental factors, may help models capture more variance. In hindsight, performing deeper data preprocessing, including feature engineering and normalization, could enhance model learning. Lastly, techniques such as hyperparameter tuning and cross-validation may provide more robust results, alongside deploying deep learning models such as LSTM and CNN can be deemed more useful when capturing temporal dependencies related to the targets.

**References**

[1] A. Cincinelli and T. Martellini, ‘Indoor Air Quality and Health’, International Journal of Environmental Research and Public Health, vol. 14, no. 11, 2017.

[2] K. W. Tham, ‘Indoor air quality and its effects on humans—A review of challenges and developments in the last 30 years’, Energy and Buildings, vol. 130, pp. 637–650, 2016.