Air quality dataset from an indoor airport travelers transit area

Prepared by,

Sayma Sultana (2021-3-60-105)

Moinul Islam (2021-3-60-241)

Department of Computer Science and Engineering(CSE), EWU

Submitted to,

Dr. Mohammad Rifat Ahmmad Rashid

Associate Professor, Dept of Computer Science and Engineering, EWU

Submission Date: January 25, 2025

Abstract

Air pollution has severe effects on the environment, it has become essential to monitor this for environmental and public health concerns. This experimental dataset has been collected over 1 year (October 2022-October 2023) in an indoor travelers' transit area in the Brindisi airport, Italy. The data was collected using three Internet of Things (IoT) sensing nodes equipped with Raspberry Pi 4 (as processing unit) and three low-cost commercial sensors (namely: Adafruit MiCS5524, Sensirion SCD30, Sensirion SPS30) are connected. Each sensing node sampled air parameters every 2 seconds, resulting in over 7 million data Each collected record composing the experimental dataset contains (i) the identifier of the IoT node that sampled the air parameters; (ii) the presence of gases (as a unified value concentration); (iii) the concentration of carbon dioxide (CO_2) in the travelers' transit area, together with air temperature and humidity; and (iv) the concentration of particulate matter (PM) in the indoor monitored environment – in terms of particles' mass concentration ($\mu g/m^3$), number of particles ($\#/cm^3$), and typical particle size (µm) – for particles with a diameter up to 0.5 µm ($PM_{0.5}$), 1 µm (PM_1), 2.5 µm ($PM_{2.5}$), 4 µm (PM_4), and 10 µm (PM_{10}). We have applied several machine learning models such as Xgboost, neural network, linear regression, and random forest regressor, we compared their performance and found that Xgboost got the highest accuracy which was 81 percent. We have also performed a t-test (paired t-test) to see if there is any significant relationship between the dataset variables. ANOVA was also performed to see if there was any difference between the mean of the two groups or conditions. The present research provides several lessons learned in air quality monitoring using IoT-based systems and lays the foundation for developing data-driven solutions to reduce indoor air pollution and improve public health outcomes.

Table of Contents

List of Figure

List of Table

Chapter 1 Introduction

1.1 Dataset Overview

This article presents a dataset collected from Science Direct as part of the European project InSecTT, containing air quality parameters sampled in the travelers' transit area of "Aeroporto del Salento" in Brindisi, southern Italy. The dataset, referred to as “complete_measure_node_airport_gold”, was gathered using prototypical IoT sensing nodes developed at the IoTLab of the University of Parma. The sampling was conducted over one year, from October 2022 to October 2023, within the indoor transit area of Brindisi airport. It contains 7M data. Each monitoring device deployed for data collection was equipped with a Raspberry Pi 4 (RPi4) Single Board Computer (SBC) as its processing unit, along with three low-cost commercial sensors: Adafruit MiCS5524, Sensirion SCD30, and Sensirion SPS30. The dataset supports the estimation of daily air quality variations by analyzing traveler flow trends. It is a valuable resource for researchers, program managers, operators, and airport administrators, enabling the use of statistical methods, Machine Learning (ML), and Deep Learning (DL) techniques to analyze single or combined air quality parameters and identify correlations. Moreover, researchers can utilize the dataset to compare air quality data from similar indoor environments (e.g., airports, offices, homes) and outdoor contexts, aiding in the assessment of air quality variability. The dataset enables a wide range of analytical approaches, including statistical techniques and ML/DL algorithms, to extract meaningful insights.

1.2 The variable of the dataset & its parameter

Id_measure: Unique identifier of the record (as an integer value)

Node_id: Textual identifier of the sampling IoT device

Ads1115_value: Gases concentration value (as an aggregate value) returned by the MiCS5524 sensor

Ads1115_voltage: Voltage value corresponding to the gas concentration identified by the “ads1115_value” field

Scd30_co2: CO2 concentration, expressed as parts per million (ppm)

Scd30_temp: Air temperature, expressed as Celsius degrees (°C)

Scd30_hum: Air humidity, expressed as a percentage per relative humidity (%RH)

Sps30_pm05_count: Spatial concentration (expressed as #/cm3) of particles with a diameter up to 0.5 µm (PM0.5)

Sps30_pm1_count: Spatial concentration (expressed as #/cm3) of particles with a diameter up to 1 µm (PM1)

Sps30_pm1_ug: Mass concentration (expressed as µg/m3) of particles with a diameter up to 1 µm (PM1)

Sps30_pm25_count: Spatial concentration (expressed as #/cm3) of particles with a diameter up to 2.5 µm (PM2.5)

Sps30_pm25_ug: Mass concentration (expressed as µg/m3) of particles with a diameter up to 2.5 µm (PM2.5)

Sps30_pm4_count: Spatial concentration (expressed as #/cm3) of particles with a diameter up to 4 µm (PM4)

Sps30_pm4_ug: Mass concentration (expressed as µg/m3) of particles with a diameter up to 4 µm (PM4)

Sps30_pm10_count: Spatial concentration (expressed as #/cm3) of particles with a diameter up to 10 µm (PM10)

Sps30_pm10_ug: Mass concentration (expressed as µg/m3) of particles with a diameter up to 10 µm (PM10)

Sps30_pm_typ: Typical particle size (expressed as µm) of the PM

Ts_insertion: Timestamp (in Unix format) of the data sampling's time instant

1.3 Domain Interest

The domain of the present dataset is the environmental monitoring of an airport, focusing on air quality analysis. Air pollution affects public health and the environment therefore it primarily needs to be monitored and analyzed. The data here are of high resolution, collected from IoT-based sensors in a travelers' transit area over a year, thus allowing a detailed insight into indoor air pollution patterns. A similar analysis might have the potential to enhance air quality management by helping policymakers design proper interventions that meet public health concerns with the help of ML models. The sc30_temp-temperature readings from the Sensirion SCD30 sensor serve as the project's target measurement. Real temperature measurements help scientists understand better how different environmental conditions affect air quality substances. The following models have been applied: We tested Linear Regression plus two advanced models known as Random Forest and Neural Networks. Multiple methods can produce sc30_temp models based on these sets. These approaches help us better understand the overall air quality environment and temperature patterns. The analysis of this dataset should be helpful in:

- These findings will help improve how we deal with air quality within indoor transport spaces.
- Our models help environmental monitoring systems predict changes in air quality.

- Instant measures should be taken to help travelers stay safe and at ease.

Chapter 2 Hypothesis Setting

The heatmap reveals both significant matching values between sps30 particles and parallel correlations between ads1115_value data with ads1115_voltage measurements. Intermediate levels of statistical connection link scd30_temp with scd30_hum yet most other variables show little to no correlation. Through correlation assessment, this method can recognize linked variables that direct the selection of important characteristics.

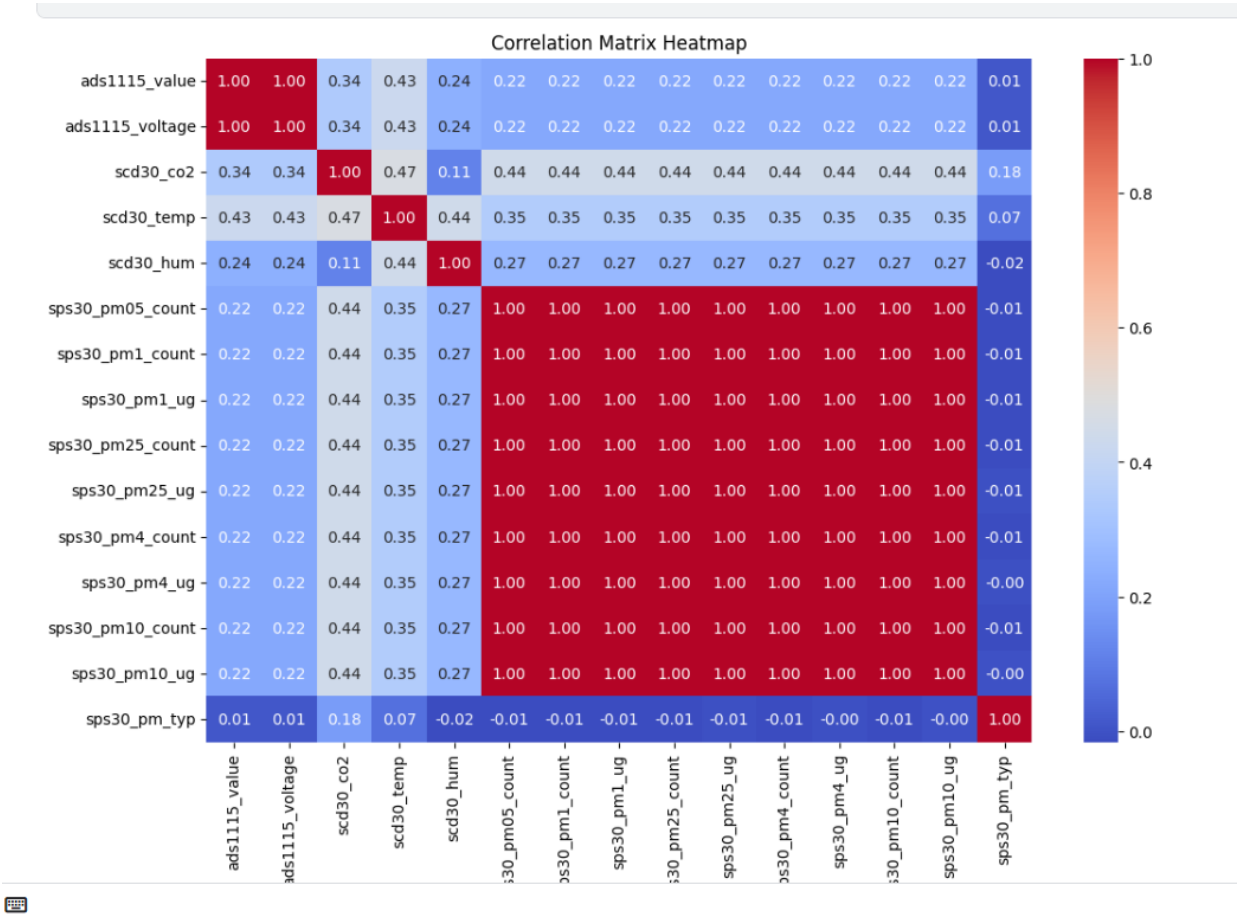


Figure 1: Correlation Heatmap

2.1 Hypothesis 1

Air quality degrades during peak travel times: Higher foot traffic from travelers increases CO2 and particulate matter due to more human activity.

2.2 Hypothesis 2

CO2 concentration correlates with the number of people in the transit area: More people passing through the area will increase CO2 levels, as they exhale.

2.3 Hypothesis 3

PM concentration varies with proximity to sources of particulate matter: Areas with more movement (e.g., luggage trolleys, passengers) may show higher PM levels, especially for smaller particles.

2.4 Hypothesis 4

Temperature and humidity are inversely related to PM levels: Higher humidity may cause particles to settle, while higher temperatures could disperse particles or increase the chemical reactions that produce them.

2.5 Hypothesis 5

The gas concentration is higher near sources of pollution: Areas close to airport facilities, vehicle exhaust, or food service areas may have higher levels of gases like CO, ethanol, or methane.

Chapter 3 Statistical Testing

3.1 Statistical Test

We have used a t-test to see if there is any statistical significance between the means of the two variables.

T-Test (Paired T-Test): We took CO2 concentration (scd30_co2) and Temperature (scd30_temp) as numerical columns in the dataset.

3.1.1 Hypothesis

a. Null Hypothesis (H_0): There is no significant difference between the mean CO2 concentration and the mean Temperature.
b. Alternative Hypothesis (H_1): There is a significant difference between the mean CO2 concentration and the mean Temperature.

3.1.2 Outcome

The p-value represents the probability of observing the difference between the means under the null hypothesis. The p-value < 0.05, we reject the null hypothesis, indicating a significant difference between the two means (CO2 concentration and Temperature).

Table 1: Statistical hypothesis

No.	Null Hypothesis (H_0)	Alternative Hypothesis (H_1)	P-value	α	Analysis	Result
H1o	No significant difference between the mean CO2 concentration and mean temperature.	There is a significant difference between the mean CO2 concentration and mean temperature.	0.00	0.05	P-value $< \alpha$	Reject the null hypothesis.

3.2 ANOVA

In this analysis, the goal is to examine the CO2 concentration (scd30_co2) across different days of the week, which is a categorical variable(ts_insertion).

3.2.1 Hypotheses:

a. Null Hypothesis (H_0): There is no significant difference in CO2 concentration across the days of the week.

b. Alternative Hypothesis (H_1): There is a significant difference in CO2 concentration across the days of the week.

3.2.2 Outcome:

Reject the null hypothesis, indicating a significant difference in CO2 levels across the days.

Table 2: One-way ANOVA

Source	df	sum_sq	mean_sq	F	PR(>F)
Days of the Week	6	640.34	106.72	0.59	0.735
Residual	343	61585.93	179.55	NaN	NaN

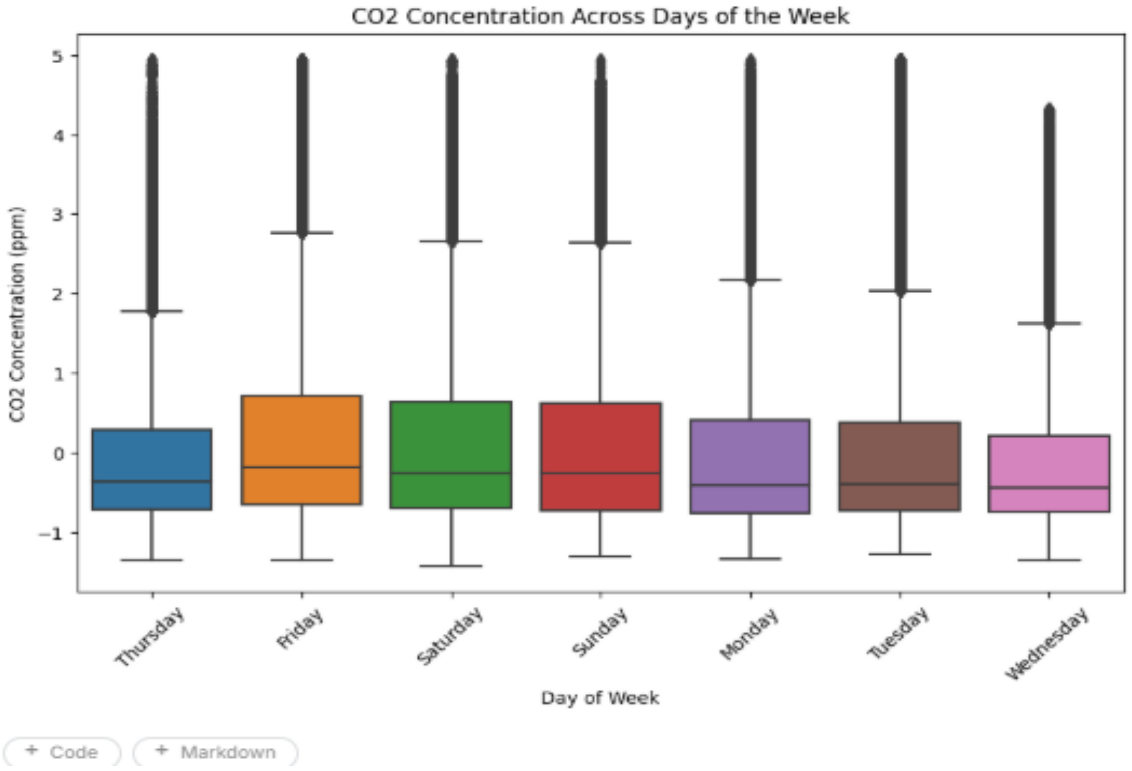


Figure 2: ANOVA Box Plot

3.3 Regression Analysis

The code implements a linear regression model to find a significant relationship between the dependent and independent variables in the dataset, it helps to establish a relationship between variables by fitting a straight line. It is widely used because of its simplicity and interpretability. It begins by splitting the data into training and test sets using train_test_split (), where the training set is used to train the model, and the test set is used for evaluation. The model is then trained using the. fit () method on the training data (X_train, y_train), where it learns the relationship between the features and the target variable by fitting a linear equation. Once the model is trained, it makes predictions on the test data (X_test) using the. Predict () method, and the predicted values are stored in y_pred_lr. To evaluate the model's performance, Mean Squared Error (MSE**)** is calculated to measure the average squared difference between the predicted and actual values, with a lower MSE indicating better accuracy. Additionally, R-squared (R^2) is computed to assess how well the model explains the variance in the target variable, with values closer to 1 indicating a good fit. Finally, the code prints the first five predicted values, along with the MSE and R^2 scores, to evaluate the model's performance.

Chapter 4 Machine Learning Test

4.1 Data Preprocessing

Before applying machine learning models, we found the dataset was not in the correct order with 7 million data, so we used a delimiter keyword, then we put all the column names in the array list and used the set to reorder them. Then we saw some missing values using the (isnull) function to see if there were any missing values, we saw some values were missing so we used mean for numerical value and mode for categorical value to fill up the null values also we dropped id measure before we fill up the null values. We searched for if there was any outlier and many outliers were found in the dataset, then to show the dataset with the outlier removed properly we used standardized scaling to show the outlier removed plot. Then we have shown correlation matrix and heatmap to know how the rows and columns are co-related with each other. The values with 1 are highly correlated and 0 are moderate and -1 are low correlated.

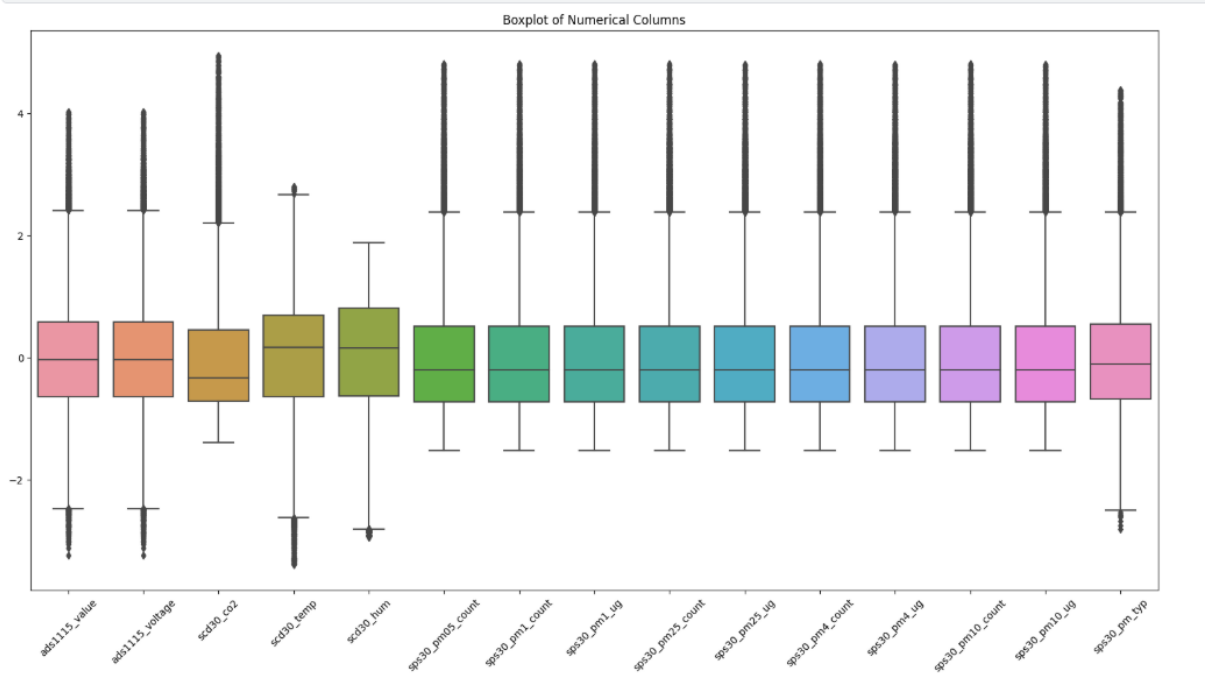


Figure 3: After Removing Outlier

4.2 Model Selection

Our selection of computer learning models matched our goal to work with both varied air quality information and make accurate predictions. We have used **Linear Regression** as its starting point since it helps show basic relationships between features and the temperature readings (sc30_temp). Since air quality data contains irregular patterns and variable interactions

standard models didn't fully capture its features. **Random Fore**st excels in finding hidden connections between air quality data because it needs minimal preparation work on input information. The model handles noise well while revealing which air quality metrics have the strongest impact on pollution levels. Our project team picked **XGBoost** because it handles big datasets quickly and produces precise prediction results. The technique reduces errors step by step while preventing overfitting which makes it suitable for large datasets of this size (7 million records). Our **Neural Network** analysis aims to reach the maximum potential of our available information. Their deep learning architecture processed the extensive dataset better than basic models and uncovered vital relationship patterns that other models would have missed.

Our testing of multiple models across the dataset showed how to find the best method for reliable air quality forecasting.

4.3 Training and Testing

We have split the dataset into two portions, one is for training and the other is for a test. **30** percent of the data was used for testing and **70** percent was used for training. After running the machine learning models, the first model applied was random forest as the targeted variable was regression we used r-squared to show the accuracy of the model, which was 73 percent. Next, we applied Xgboost which gave an accuracy of 81 percent, after that linear regression was applied which gave 43 percent accuracy and the other one was neural network which was 70 percent. From the four models, the highest accuracy was obtained from Xgboost.

4.4 Explainable AI Techniques

Through Explainable Artificial Intelligence (XAI) we can understand exactly how machine learning models make their decisions. Human users can trust how ML software makes its choices when they understand its inner workings models. As deep learning systems become more advanced their inner workings become harder to understand.

Their ability to track how models decide diminishes. XAI aims to:

- Show users specific details about how models make their decisions.
- People need to trust AI systems and we achieve trust by explaining model results.
- Need testing for bias to help them work better for everyone.
- Simplify the work to test and optimize machine learning systems.

We have used both SHAP and LIME to show the overall feature importance across all the predictions and focus on the contribution of features of specific features. The SHAP plot has been used to show the highest importance of features that have an impact with the model.

LIME will make slight changes in the input to understand how changing data affects model prediction, so it helps determine which features affect the model more. Through this process, LIME helps feature identification that the instance drives the decision of the model for a particular instance. Instead, LIME provides an answer regarding which understanding of exactly a certain prediction was made in a model through local explanations; it makes such complex models even more transparent and interpretable.

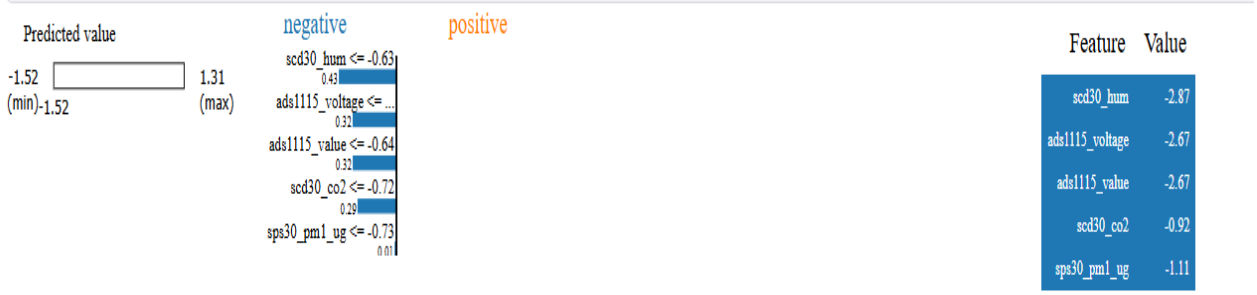


Figure 4: LIME Plot

The methods help stakeholders understand and prove that the model shows correct choices. They both aid in finding feature problems and correcting of biased model parameters. When one special feature determines model decisions, it requires careful assessment and possibly change. By seeing the plots of the SHAP and lime we can see which feature impacts to pull the

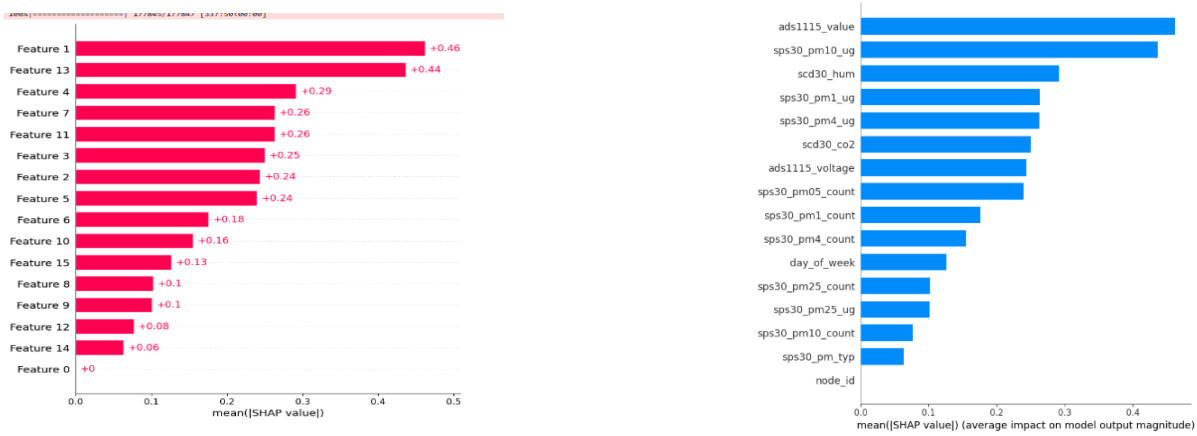
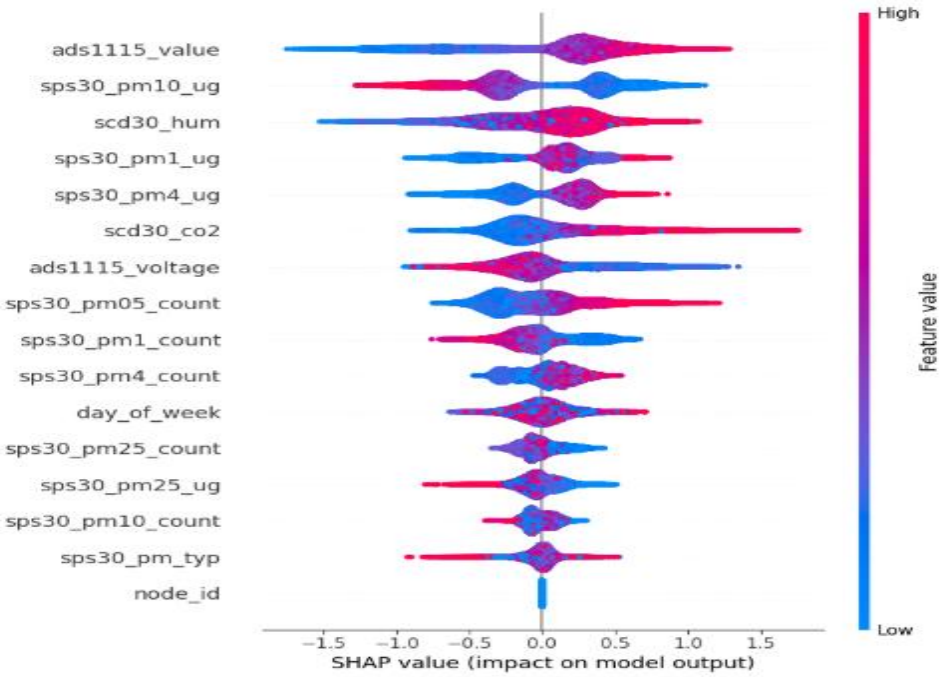


Figure 5: SHAP Bar Plot

predicted value lower or higher. In our project, the scd30_hum, ads1115_voltage, ads1115_value, scd30_co2, and sps30_pm1_ug these features pull the prediction towards negative after visualizing the plot of LIME. In SHAP there were three plots, the first plot summary exhibits average SHAP values by feature. The ranking system arranges features according to their general average effect on model prediction outcomes. Then the second one displays feature importance specifically based on SHAP values while also providing feature identification based on mapping dataset names. The violin plot shows the density of the impact, wider sections of the violin plot indicate that much data falls within that range.

Figure 6: SHAP Violin Plot



4.5 Model Insights

The model outcome is highly predictive, relying on two essential features of wide-range distribution: ads1115_value and sps30_pm10_ug. The feature node_id bears a basic stock of zero, which is low in impacting the prediction accuracy of the model. The model predictions increase when sps30_pm10_ug takes high values, in the red distribution, but decrease when it takes lower values in the blue distribution. In our initial hypothesis, we assumed that the target variable would have significant influence on scd30_hum, sps30_pm10_ug, ads1115_value, and other variables such as node_id, sps30_pm_type have less impact with the target variable. ads1115_value which is gas concentration has the highest impact on the target variable.

Chapter 5 Conclusion

Multiple features including ads1115_value together with sps30_pm10_ug and scd30_hum significantly affected predictions for scd30_tmp. The observation confirmed scientific environmental reasoning regarding the relationship between temperature and these identified variables. The SHAP values allowed us to verify the feature importance while proving that ads1115_value would influence model predictions significantly. Between statistical analysis and machine learning, explainable AI worked together to discover important patterns resulting in improved model performance alongside improved readability. The model became more reliable and actionable through this integrated approach which provided deeper insight into its decision process.

For future improvement, hype tuning can be used for better performance, and also making new features using feature engineering can give better accuracy and more understanding of the dataset.

Chapter 6 References

[1] L. Davoli, L. Belli, and G. Ferrari, "Air quality dataset from an indoor airport travelers transit area," *Internet of Things (IoT) Lab, Department of Engineering and Architecture, University of Parma, Parma 43124, Italy,* pp. 1-5, 2024.

Chapter 7 Appendix: Code Analysis

Used to reorder the dataset:

```
# Load the CSV file using the correct delimiter
df = pd.read_csv(file_path, delimiter=";")

# Print the columns in their current order
print("Current Column Order:")
print(list(df.columns))

# Define the correct column order
correct_order = [
    "id_measure", "node_id", "ads1115_value", "ads1115_voltage", "scd30_co2",
    "scd30_temp", "scd30_hum", "sps30_pm05_count", "sps30_pm1_count",
    "sps30_pm1_ug", "sps30_pm25_count", "sps30_pm25_ug", "sps30_pm4_count",
```

```
    "sps30_pm4_ug", "sps30_pm10_count", "sps30_pm10_ug", "sps30_pm_typ",
    "ts_insertion"
]

# Check if all expected columns are present
if set(correct_order) == set(df.columns):
    # Reorder the DataFrame
    df = df[correct_order]
    print("Data reordered successfully.")
else:
    # Detect missing or extra columns
    missing_columns = [col for col in correct_order if col not in df.columns]
    extra_columns = [col for col in df.columns if col not in correct_order]
    print("Error: Missing or extra columns detected.")
    if missing_columns:
        print("Missing Columns:", missing_columns)
    if extra_columns:
        print("Extra Columns:", extra_columns)
```

Xgboost model that shown highest accuracy:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from xgboost import XGBRegressor


xgb_model = XGBRegressor(use_label_encoder=False, eval_metric='rmse')
xgb_model.fit(X_train, y_train)

y_pred_xgb = xgb_model.predict(X_test)
r2 = r2_score(y_test, y_pred_xgb)  # Calculate R-squared instead
print(f"Xgboost R-squared on Sample: {r2:.2f}")  # Print R-squared
```