The International Journal of Embedded Computer Engineering (IJEmCE), Vol. 1, No. 1, April 2026: 1–51

https://ejurnal.ung.ac.id/index.php/IJEmCE

Air Quality Analysis Using the Pipeline Method (Case Study: Italy Air Quality Dataset 2004–2005)

Moh. Iqbal Hunowu[1*], Rahmad Hidayat Dongka[2], Ulfatun Nadifa[3], Ade Irawaty Tolago[4]

Computer Engineering, Universitas Negeri Gorontalo, Indonesia[1,3]

Electrical Engineering, Universitas Negeri Gorontalo, Indonesia[2,4]

* rahmatdongka@ung.ac.id

Abstract -- This study aims to demonstrate how to analyze data from an air pollution dataset recorded in Italy between 2004 and 2005. During this period, air pollution occurred due to various chemical compounds present in the atmosphere. The analysis was carried out using several methods commonly applied in raw data processing, such as data normalization, data separation, and data visualization. The Air Quality dataset, obtained from the UCI Machine Learning Repository, contains 9,357 records with 15 columns representing pollutant measurements. The Pipeline method from the Scikit-learn library was used to clean, process, and transform the data in a structured manner. The results of the analysis indicate that the sensors used in the dataset have a high correlation with laboratory reference measurements (ground truth), particularly the PT08.S1(CO) sensor, which shows a correlation of 0.93 with CO(GT). The correlation heatmap visualization reveals a strong relationship between sensor readings and chemical compounds in the air, indicating that sensor data can be effectively used to monitor air quality in real time.

Keywords:
Air Quality;
Pipeline Method;
Correlation;
Pollution Sensor;
Data Visualization;

Article History:
Received: March 1, 2026
Revised: April 15, 2026
Accepted: April 16, 2026
Published: April 30, 2026

I. INTRODUCTION

Air quality is one of the most important global environmental issues because it is directly related to human health and the sustainability of urban ecosystems [1], [2]. Air pollution caused by hazardous gases such as carbon monoxide (CO), nitrogen oxides (NOx), and volatile organic compounds can increase the risk of respiratory diseases, cardiovascular disorders, and various other health problems [3], [4]. Therefore, monitoring and analyzing air quality have become essential components of environmental management and data-driven decision-making by governments and researchers [5].

The development of environmental sensor technology and the Internet of Things (IoT) has enabled the continuous collection of air quality data in large volumes from various monitoring locations [6], [3]. These data are generally obtained from multisensor devices capable of detecting concentrations of various air pollutants as well as other environmental variables periodically [2]. The availability of large datasets provides opportunities for deeper data analysis to understand air pollution patterns and to predict future air quality conditions [4], [7].

One of the datasets frequently used in air quality analysis research is the Air Quality Dataset Italy 2004–2005, which is available in an open machine learning repository. This dataset contains approximately 9,358 hourly observations collected from chemical multisensor devices deployed in urban areas with high pollution levels in Italy. The data include sensor responses to several pollutants such as carbon monoxide (CO), benzene (C6H6), nitrogen oxides (NOx), and nitrogen dioxide (NO_2), measured simultaneously with reference data from certified analyzers. This dataset has become one of the benchmark datasets widely used in data-driven air quality analysis research [8], [4].

In recent years, machine learning–based approaches have been increasingly used to analyze and predict air quality because they are capable of modeling complex relationships among various environmental variables [9]. Several studies have shown that machine learning algorithms such as Random

Forest, Gradient Boosting, and ensemble models can produce more accurate air quality predictions compared to conventional statistical methods [2], [7]. In addition, machine learning approaches are able to capture temporal patterns and nonlinear relationships in air pollution data that are often difficult to analyze using traditional methods [5].

Nevertheless, air quality data analysis does not only depend on the prediction model used but also on systematic data processing procedures. Environmental sensor data often contain various issues such as missing values, noise, and sensor drift, which require appropriate data preprocessing before further analysis can be conducted [3], [6]. Therefore, an approach that can integrate various stages of data analysis in a structured and consistent manner is needed.

One approach widely used in modern data analysis is the pipeline method, which is an analytical workflow that integrates data collection, data cleaning, feature transformation, and modeling processes into a single organized system [10]. The pipeline approach allows each stage of analysis to be performed automatically and repeatedly, thereby improving efficiency, consistency, and research reproducibility [3]. Moreover, pipelines facilitate the development and evaluation of machine learning models when analyzing complex environmental datasets [4].

Based on this background, this study aims to analyze air quality using the pipeline method by utilizing the Air Quality Italy 2004–2005 dataset as a case study. This research is expected to provide an overview of systematic air quality data analysis stages, starting from data preprocessing to model development and evaluation. The results of this study are expected to contribute to the development of more effective environmental data analysis methods and support efforts to monitor and control air pollution in urban areas.

II. METODE

This study employs a quantitative approach based on data analysis to examine the relationships among air pollutant variables in the Air Quality Italy 2004–2005 dataset. The dataset used in this research is obtained from the UCI Machine Learning Repository, which contains 9,357 observation records with 15 variables, including air pollutant measurements, sensor data, and environmental variables such as temperature and humidity.

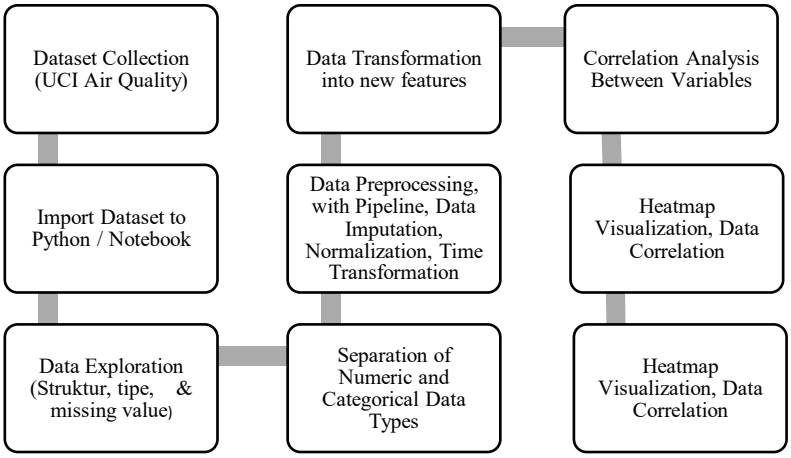

FIGURE 1. Research Flow Diagram

The data analysis procedures used in this study are described as follows.

1) Notebook and Dataset Preparation:

The first step is to log in to the Kaggle platform and create a new notebook. The Air Quality dataset is then added to the notebook using the provided link.

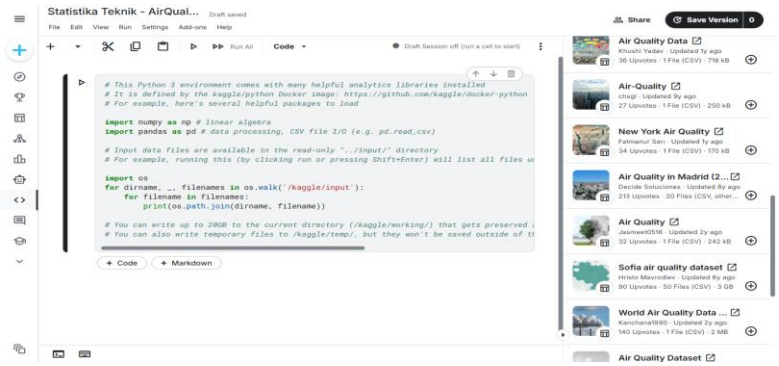

FIGURE 2. Kaggle Notebook Display

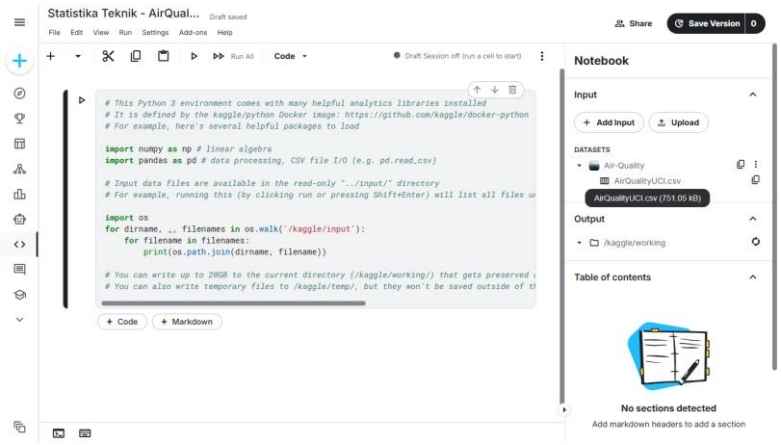

FIGURE 3. Adding Air Quality Dataset to Notebook

Based on the figures above, it can be seen that the "Air Quality" dataset has been successfully added and is ready for analysis.

2) Dataset Reading and Exploration:

After the dataset is loaded, the next step is to access and explore it. Add a new code cell by clicking "+code" and enter the following syntax:

```
fpath = "/kaggle/input/airquality/AirQualityUCI.csv"
base  = pd.read_csv(fpath)
base.info()
```

FIGURE 4. Initial Dataset Reading Syntax

The explanation of the above syntax is as follows. The fpath variable stores the dataset location in the Kaggle file system. The base variable stores the dataset content. The pd.read_csv() function is a built-in Pandas function used to read .csv format files. Meanwhile, base.info() is used to display general information about the dataset.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9357 entries, 0 to 9356
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Date            9357 non-null   object
 1   Time            9357 non-null   object
 2   CO(GT)          9357 non-null   float64
 3   PT08.S1(CO)     9357 non-null   int64
 4   NMHC(GT)        9357 non-null   float64
 5   C6H6(GT)        9357 non-null   float64
 6   PT08.S2(NMHC)   9357 non-null   int64
 7   NOx(GT)         9357 non-null   int64
```

```
 8   PT08.S3(NOx)    9357 non-null   int64
 9   NO2(GT)         9357 non-null   int64
10   PT08.S4(NO2)    9357 non-null   int64
11   PT08.S5(O3)     9357 non-null   float64
12   T               9357 non-null   float64
13   RH              9357 non-null   float64
14   AH              9356 non-null   float64
dtypes: float64(7), int64(6), object(2)
memory usage: 1.1+ MB
```

FIGURE 5. Output of Initial Dataset Reading Syntax

From the output above, it can be determined that the dataset has 9,357 rows of data and 15 columns. The Date and Time columns are object type because the data contains numbers and special symbols. The following are abbreviation descriptions for each column in the dataset:

- CO(GT): Original measurement value of Carbon Monoxide from laboratory reference instrument.
- PT08.S1(CO): Measurement value of Carbon Monoxide from sensor.
- NMHC(GT): Original measurement value of Non-Methane Hydrocarbons from laboratory reference instrument.
- C6H6(GT): Original measurement value of Benzene from laboratory reference instrument.
- PT08.S2(NMHC): Measurement value of Non-Methane Hydrocarbons from sensor.
- NOx(GT): Original measurement value of Nitrogen Oxide from laboratory reference instrument.
- PT08.S3(NOx): Measurement value of Nitrogen Oxide from sensor.
- NO2(GT): Measurement value of Nitrogen Dioxide from laboratory reference instrument.
- PT08.S4(NO2): Measurement value of Nitrogen Dioxide from sensor.
- PT08.S5(O3): Measurement value of Ozone from sensor.
- T: Temperature measurement value.
- RH: Relative Humidity measurement value.
- AH: Absolute Humidity measurement value.

3) Missing Value Inspection:

After understanding the data types used in the dataset, the next step is to check whether there are empty data (missing values) in each column.

```
base.isna().sum()
```

FIGURE 6. Missing Value Inspection Syntax

```
Date           0
Time           0
CO(GT)         0
PT08.S1(CO)    0
NMHC(GT)       0
C6H6(GT)       0
PT08.S2(NMHC)  0
NOx(GT)        0
PT08.S3(NOx)   0
NO2(GT)        0
PT08.S4(NO2)   0
PT08.S5(O3)    0
T              0
RH             0
AH             1
dtype: int64
```

FIGURE 7. Missing Value Inspection Output

The output above shows that almost all columns in the dataset have no empty rows, except the AH (Absolute Humidity) column which has 1 missing value. It should be noted that this check only detects NaN values, unlike error values which are still considered filled.

4) Data Type Separation:

Next, a separation is performed between numeric data type columns (float and int) and object data type columns using the following syntax:

```
num cols =
base.select_dtypes(include=['int64','float64']).columns
cat_cols = base.select_dtypes(include=['object']).columns
print(f"num_col: {num_cols}")
print(f"cat_col: {cat_cols}")
```

FIGURE 8. Data Type Separation Syntax

```
num_col: Index(['CO(GT)', 'PT08.S1(CO)', 'NMHC(GT)',
'C6H6(GT)',
        'PT08.S2(NMHC)', 'NOx(GT)', 'PT08.S3(NOx)',
'NO2(GT)',
        'PT08.S4(NO2)', 'PT08.S5(O3)', 'T', 'RH', 'AH'],
        dtype='object')
cat_col: Index(['Date', 'Time'], dtype='object')
```

FIGURE 9. Data Type Separation Output

The num_cols variable stores int and float type columns from the base dataframe. Similarly, cat_cols stores object type columns. The output shows 13 numeric columns and 2 object type columns, namely Date and Time.

5) Data Cleaning with Pipeline Method:

After data separation, the next step is to clean the data using the Pipeline method. Pipeline is a method used to clean, process, model, and visualize data in a structured and efficient manner.

```
import pandas as pd
import numpy as np
from sklearn.pipeline      import Pipeline
from sklearn.compose       import ColumnTransformer
from sklearn.preprocessing import (OneHotEncoder,
StandardScaler,
                                   FunctionTransformer)
from sklearn.impute        import SimpleImputer
```

FIGURE 10. Import Library Syntax for Pipeline Data

```
# ===== 1. COLUMNS =====
num_cols  =
['CO(GT)','PT08.S1(CO)','NMHC(GT)','C6H6(GT)',

'PT08.S2(NMHC)','NOx(GT)','PT08.S3(NOx)','NO2(GT)',
            'PT08.S4(NO2)','PT08.S5(O3)','T','RH','AH']
date_cols = ['Date']
time_cols = ['Time']
```

FIGURE 11. Variables for Storing Column Data

```
# ===== 2. DATE & TIME TRANSFORMER =====
def extract_date(X):
    base = pd.DataFrame(X, columns=['Date'])
    base['Date'] = pd.to_datetime(base['Date'],
errors='coerce')
    out = pd.DataFrame({
        'year':      base['Date'].dt.year.fillna(0),
        'month':     base['Date'].dt.month.fillna(0),
        'day':       base['Date'].dt.day.fillna(0),
        'dayofweek':
base['Date'].dt.dayofweek.fillna(0),
        'quarter':   base['Date'].dt.quarter.fillna(0)
```

```
    })
    return out.values

def extract_time(X):
    base = pd.DataFrame(X, columns=['Time'])
    base['Time'] = base['Time'].str.replace('.', ':',
regex=False)
    base['Time'] = pd.to_datetime(base['Time'],
                        format='%H:%M:%S',
errors='coerce')
    return pd.DataFrame({
        'hour':   base['Time'].dt.hour,
        'minute': base['Time'].dt.minute,
        'second': base['Time'].dt.second
    }).fillna(0).values
```

FIGURE 12. Date and Time Object Data Transformation Syntax

```
# ===== 3. PIPELINE =====
num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler',  StandardScaler())
])
date_pipeline = Pipeline([
    ('date', FunctionTransformer(extract_date,
validate=False))
])
time_pipeline = Pipeline([
    ('time', FunctionTransformer(extract_time,
validate=False))
])
preprocessor = ColumnTransformer(transformers=[
    ('num',  num_pipeline,  num_cols),
    ('date', date_pipeline, date_cols),
    ('time', time_pipeline, time_cols)
])
```

FIGURE 13. Pipeline Data Syntax

```
# ===== 4. TRANSFORM =====
X       = base.copy()
X_trans = preprocessor.fit_transform(X)
print("Original transform shape:", X_trans.shape)

feature_names  = list(num_cols)
feature_names +=
['year','month','day','dayofweek','quarter']
feature_names += ['hour','minute','second']

print("Number of feature names:", len(feature_names))
print("Number of actual columns:", X_trans.shape[1])
X_clean = pd.DataFrame(X_trans, columns=feature_names)
print("Remaining NaN:", X_clean.isna().sum().sum())
```

FIGURE 4. Pipelined Data Transformation Syntax

```
Original transform shape: (9357, 21)
Number of feature names : 21
Number of actual columns: 21
Remaining NaN           : 0
```

FIGURE 6. Pipeline Data Output

The explanation of each code table above is as follows. Table VII is used to call the libraries required for the pipeline process. Table VIII separates columns by data type, specifically the object type (Date and Time), to avoid errors during processing. Table IX contains transformation functions to convert Date and Time object columns into several integer-type numeric columns. Table X shows the pipeline configuration for processing numeric and object data that has been separated. Table XI shows the dataset transformation process using the configured pipeline. Table XII is the output proving that the pipeline process was successful, producing 21 columns with no missing values (Remaining NaN: 0).

6) Data Visualization with Correlation Heatmap:

After the data has been processed, the next step is to visualize the data in the form of a correlation matrix (heatmap). The following syntax is used:

```
import matplotlib.pyplot as plt

base_num  = base[num_cols]
corr      = base_num.corr().abs()
threshold = 0.9
```

FIGURE 7. Matplotlib Library Import and Correlation Calculation

```
high_corr_features = corr.columns[(corr >
threshold).any()]
print("Jumlah fitur berkorelasi tinggi:",
len(high_corr_features))
print(high_corr_features.tolist())

corr_high =
X_clean[high_corr_features].corr().round(2)
plt.figure(figsize=(10, 8))
plt.imshow(corr_high)
plt.colorbar()
plt.title("High Correlation Features")
```

FIGURE 8. Konfigurasi Fitur Berkorelasi Tinggi

```
plt.xticks(range(len(corr_high.columns)),
           corr high.columns, rotation=90,
fontsize=8)
plt.yticks(range(len(corr_high.columns)),
           corr_high.columns, fontsize=8)
for i in range(len(corr_high.columns)):
    for j in range(len(corr_high.columns)):
        plt.text(j, i, corr_high.iloc[i, j],
                 ha="center", va="center",
fontsize=7)
plt.tight_layout()
plt.show()
```

FIGURE 9. Konfigurasi Label Sumbu

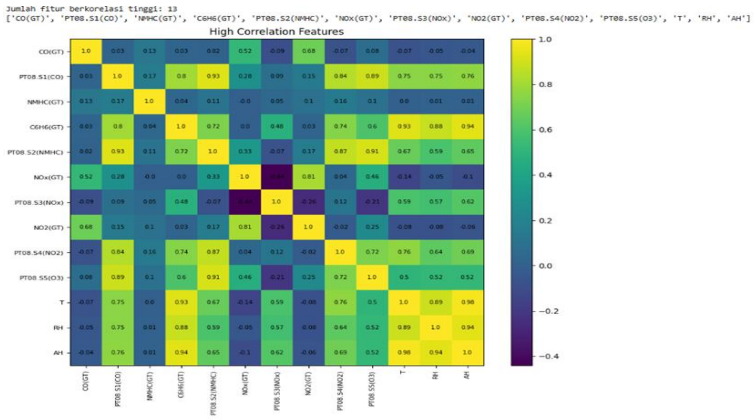

FIGURE 10. Output Heatmap/Correlation Matrix Between Features

The results indicate that several sensors show strong correlations with laboratory reference measurements (ground truth), suggesting that the multisensor system is capable of representing pollutant concentrations with a high degree of reliability. In particular, the PT08.S1(CO) sensor exhibits a very strong correlation with CO(GT) with a coefficient of 0.93, indicating its effectiveness in detecting carbon monoxide concentrations. Additionally, environmental variables such as temperature, relative humidity, and absolute humidity were found to influence pollutant concentrations, highlighting the importance of meteorological factors in air quality analysis. The implementation of a pipeline-based data processing approach also improves the efficiency, consistency, and reproducibility of data analysis, demonstrating its

potential for supporting data-driven air quality monitoring systems and predictive environmental models in urban environments.

III. RESULTS AND DISCUSSION

A. Results

Based on the results of the analysis conducted, the Air Quality dataset indicates that several sensors have a relatively strong relationship with laboratory reference measurements. A high correlation between the sensor readings and the reference values suggests that the sensors are capable of representing air pollutant concentrations with reasonable accuracy. Previous studies have also shown that the use of multisensor systems in air quality monitoring can provide pollutant concentration estimates that are close to laboratory measurement values when appropriate data analysis methods are applied [1], [10].

B. Discussion

The results of the correlation analysis show that the PT08.S1(CO) sensor has a very high correlation with the CO(GT) value, with a correlation coefficient of 0.93. This high correlation indicates that the sensor is capable of detecting carbon monoxide concentrations with a relatively high level of accuracy. These findings are consistent with previous studies suggesting that multisensor-based gas sensors can effectively be used to monitor air pollutant concentrations when proper calibration and data analysis procedures are applied [1], [10].

In addition to carbon monoxide, other pollutants such as NOx and NO2 also exhibit significant relationships with their corresponding sensors. Both pollutants are classified as hazardous air pollutants that may cause adverse health effects when present at high concentrations over extended periods [2], [15]. Environmental factors such as temperature, relative humidity (RH), and absolute humidity (AH) also influence pollutant concentrations in the air. Meteorological conditions can affect pollutant dispersion, chemical reactions, and accumulation processes in the atmosphere.

Therefore, these environmental variables should be considered in comprehensive air quality analysis [11], [12]. Furthermore, the use of the pipeline method in the data analysis process provides advantages in terms of efficiency and consistency in data processing. The pipeline approach enables automated and integrated data cleaning, feature transformation, and data normalization, thereby facilitating the analysis of complex datasets. This approach has been widely applied in machine learning–based research to improve data processing quality and enhance reproducibility in scientific studies [13], [14].

IV. CONCLUSION

Based on the analysis conducted on the Air Quality dataset from Italy during 2004–2005, the following conclusions can be drawn.

First, by using sensor data and its correlation with chemical compounds, we can accurately estimate the percentage content of various pollutants in the air. Sensors with high correlation to specific chemical compounds are very helpful in monitoring air quality and measuring pollutant concentrations.

Second, the applied Pipeline method successfully performed data cleaning, transformation, and normalization efficiently. The dataset, which originally had heterogeneous data types (object and numeric), was successfully converted into a consistent format ready for further analysis.

Third, the correlation matrix visualization (heatmap) shows strong relationships between certain sensors and the measured chemical compounds, enabling this sensor data to be used to measure and monitor in real-time what percentage of chemical compounds are in the air. This is very important for environmental health evaluation and appropriate policy-making regarding air pollution control.

V. REFERENCES

[1] Khan, A., Rahman, M., & Chen, Y. (2024). Data pipeline architectures for environmental monitoring analytics. Journal of Environmental Informatics, 43(2), 145–158.

[2] Singh, D., & Kumar, V. (2023). Machine learning applications in air pollution analysis. Environmental Monitoring and Assessment, 195(3), 1–15. https://doi.org/10.1007/s10661-023-11025-4

[3] Rahman, M., Hasan, M., & Karim, A. (2022). Data preprocessing techniques for environmental sensor datasets. Sensors, 22(15), 5678. https://doi.org/10.3390/s22155678

[4] Zhang, Y., Li, H., & Wang, X. (2023). Data-driven air quality prediction using ensemble learning models. Atmospheric Environment, 296, 119553. https://doi.org/10.1016/j.atmosenv.2023.119553

[5] Agbehadji, I. E., & Obagbuwa, I. C. (2024). Machine learning and deep learning techniques for spatiotemporal air quality prediction: A systematic review. Atmosphere, 15(11), 1352. https://doi.org/10.3390/atmos15111352

[6] Kumar, P., Singh, A., & Gupta, R. (2022). Air pollution prediction using machine learning approaches: A comprehensive review. Environmental Research, 204, 112020. https://doi.org/10.1016/j.envres.2021.112020

[7] Wang, S., Li, X., & Zhao, Y. (2023). Air quality prediction using hybrid machine learning models. Environmental Pollution, 316, 120567. https://doi.org/10.1016/j.envpol.2022.120567

[8] Ashraf, M., & Moradiya, K. (2025). Machine learning–driven carbon monoxide prediction using the UCI air quality dataset. Australian Journal of Artificial Intelligence Review, 7(1), 45–59.

[9] Aram, F., García, E. H., Solgi, E., & Mosavi, A. (2024). Air quality forecasting using machine learning: Comparative analysis and ensemble strategies. Water, Air, & Soil Pollution, 235(4), 198. https://doi.org/10.1007/s11270-024-06915-3

[10] Li, J., Zhao, Z., & Chen, Y. (2021). Machine learning approaches for air pollution prediction: A review. Environmental Modelling & Software, 139, 105025. https://doi.org/10.1016/j.envsoft.2021.105025

[11] Sharma, S., & Goyal, P. (2022). Forecasting air pollutant concentration using data analytics. Sustainable Cities and Society, 77, 103553. https://doi.org/10.1016/j.scs.2021.103553

[12] Chen, Z., Zhang, T., Chen, Z., Xiang, Y., & Xuan, Q. (2021). High-resolution dataset for air quality estimation. Environmental Data Science, 1, e15. https://doi.org/10.1017/eds.2021.15

[13] Fassò, A., Rodeschini, J., Moro, A. F., & Finazzi, F. (2022). Environmental datasets for air quality monitoring and prediction. Scientific Data, 9(1), 432. https://doi.org/10.1038/s41597-022-01523-4

[14] D'Elia, I., Briganti, G., Vitali, L., Piersanti, A., Righini, G., & Ciancarella, L. (2021). Measured and modelled air quality trends in Italy. Atmospheric Chemistry and Physics, 21(14), 10825–10844. https://doi.org/10.5194/acp-21-10825-2021

[15] Blanco, G., Barco, L., Innocenti, L., & Rossi, C. (2024). Urban air pollution forecasting using machine learning and satellite observations. Environmental Data Science, 3, e8. https://doi.org/10.1017/eds.2024.8