Course Code: SAIA 2113

Course Name: Machine Learning

Title: Smart City (Air Quality Solution)

Name & Matric ID:	1. WAN ALIF DANIAL BIN WAN KAMARULFARID (A24AI0093) 2. FARIN BATRISYIA BINTI SAIPUL NIZAM (A24AI0030) 3. MUHAMMAD DANISH IQBAL BIN MOHAMAD HASSAN (A24AI0052)
Section:	SECTION 4
Lecturer's Name:	ENCIK ADAM BIN KHAIRUDDIN

ACKNOWLEDGMENT

First and foremost, we are profoundly grateful to Allah SWT and His messenger, Prophet Muhammad SAW, for granting us the strength, guidance, and blessings to complete our first assignment of this semester. It is through His grace that we were able to overcome challenges and accomplish this task, Alhamdulillah.

We would like to express our heartfelt gratitude to our parents and family members for their unwavering support throughout the semester. They have been our pillars of strength, understanding our struggles, particularly in meeting the demands of our academic workload. Their constant encouragement and practical advice have been invaluable in helping us navigate through difficulties.

We would also like to extend our deepest appreciation to our peers, whose collaboration and assistance have been crucial in completing this project. Their willingness to help and share insights has made this journey more manageable and rewarding.

A special thanks goes to Dr. Adam Bin Mohd Khairuddin, the lecturer for the Machine Learning course. His dedication, guidance, and support have been instrumental in shaping this assignment. Dr. Adam Bin Mohd Khairuddin consistently motivated us by emphasizing the importance of maximizing our marks. His thoughtful approach, including sharing resources through E-Learning and engaging us with alternative teaching methods, helped us stay on track and better understand the subject matter.

Lastly, we aspire for this task to meet the expectations and to be free of errors, reflecting the effort and dedication invested in its completion. We pray for a favourable evaluation and hope to pass this course with excellence.

Thank you to everyone who has contributed to this accomplishment.

Table of Contents

1.0 Summary

This project aims to predict urban air quality, focusing specifically on Benzene (C6H6) concentrations using machine learning techniques. The dataset used is the publicly available "Air Quality Data Set" from the UCI Machine Learning Repository, which contains over 9,000 hourly measurements of various air pollutants and meteorological data collected in Rome, Italy. Key pollutants include Carbon Monoxide (CO), Nitrogen Oxides (NOx), Nitrogen Dioxide (NO_2), and Benzene. Along with features like temperature, relative humidity, and absolute humidity. The dataset was found to be clean with no missing or duplicate values.

To build the prediction model, an Artificial Neural Network (ANN) was developed using Keras and TensorFlow. The network was designed as a Multi-Layer Perceptron (MLP) with four hidden layers, using ReLU activation functions, Batch Normalisation, and Dropout layers (set at 0.3) to prevent overfitting. The model was built with the Adam optimiser with a learning rate of 0.001 and trained across 200 epochs. The performance of the regression job was assessed using continuous output metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 score.

The trained model achieved impressive results, with a mean absolute error (MAE) of 0.336 $\mu g/m^3$, a root mean square error (RMSE) of 0.518 $\mu g/m^3$, and an R^2 value of 0.9939, indicating that the model could account for nearly 99% of the variance in benzene concentration. Visual inspection revealed consistent learning curves, low residuals, and correct predictions. A feature importance analysis revealed that PT08.S2(NMHC) was the most dominant contributor to the model's predictions, while several other features had minimal or even slightly negative influence. Overall, the experiment demonstrates how merging environmental sensor data with neural networks may result in very accurate air quality forecasts.

2.0 Introduction

I. Problem Background

The main purpose of smart city development is to provide people with a high quality of life, and one of the critical components in achieving this is maintaining clean and breathable air. However, as cities grow and more vehicles hit the roads, air pollution is becoming a bigger problem. It’s not just about the environment; poor air quality significantly impacts health by exacerbating conditions such as cardiovascular diseases and respiratory disorders, as well as contributing to skin aging (Górski et al., 2024). Pollutants like PM2.5 and NOx lead to oxidative stress and inflammation, increasing disease risk and making it harder to run cities efficiently and sustainably. This not only harms the environment but also creates major health risks for the population. Furthermore, poor air quality undermines a city's capacity to function efficiently and sustainably."

Smart cities use real-time data and advanced technologies such as Internet of Things (IoT) devices, artificial intelligence (AI), and data analytics to monitor and manage critical urban services. As noted in recent research, smart cities leverage IoT for real-time monitoring and management of urban systems, while AI supports predictive decision-making (Karthikeyan, 2024). One of the most important aspects of environmental monitoring is air quality. Integrating air quality monitoring into the smart city ecosystem allows authorities to make informed decisions, issue public warnings, and implement proactive measures to reduce pollution. However, the effectiveness of such smart projects is heavily dependent on the accuracy of the data collected. If environmental sensor data is incorrect or insufficient, it may result in false data.

The dataset from the UCI Machine Learning Repository titled **"Air Quality Data Set"** provides real-world sensor readings of various air pollutants in an Italian city. This dataset is particularly useful for research in environmental monitoring and machine learning, as it captures hourly measurements of key air pollutants and weather-related variables, providing both temporal and environmental context. More specifically, it refers to concentrations of dangerous pollutants, including:

- **Carbon Monoxide (CO):** A toxic gas primarily generated by vehicles and the incomplete combustion of fuels
- **Benzene (C6H6):** A toxic chemical used in making plastics and other products. Long-term exposure can cause serious health problems, including cancer. It is also a harmful air pollutant found in vehicle exhaust and industrial emissions.

- **Nitric Oxide (NO) and Nitrogen Dioxide (NO_2):** Both of which are nitrogen oxides, frequently produced by industrial and transportation processes, and are known to cause difficulty breathing and contribute to air pollution.

In addition to pollutant data, the dataset also provides meteorological variables that influence pollutant dispersion and behavior in the atmosphere. These include:

- **Temperature (°C):** Elevated temperatures can accelerate the rate of chemical reactions among pollutants
- **Relative Humidity (%):** Influences air quality and how particles behave
- **Absolute Humidity (g/m^3):** This is a more accurate way to measure the amount of moisture in the air

By combining pollutant levels with weather data, this dataset supports the development of predictive models for air quality. Environmental factors like temperature and humidity have a big impact on how pollutants behave. For example, higher temperatures may accelerate chemical reactions that form secondary pollutants. In the larger framework of smart cities, this combination of meteorological and environmental data supports sustainable urban planning, improves public health, and helps create a more data-driven environment. As highlighted in recent research, integrated urban services utilize meteorological and environmental data to enhance urban resilience, support public health, and facilitate informed decision-making. This data-driven approach aids in hazard preparedness, climate adaptation, and sustainable development, aligning with the UN Sustainable Development Goals (Baklanov, 2024).

II. Aim

This project aims to analyze the UCI Air Quality dataset in the context of smart city development by:

a) **Preprocessing and cleaning the dataset**

Preprocess and clean the dataset to remove common mistakes found in real-world sensor data, such as missing values, outliers, and inconsistent readings. This process assures data integrity and prepares the dataset for precise analysis and modelling, which is critical in any real-time smart city monitoring system.

b) **Exploring trends and correlations**

Analyse patterns, trends and correlations between various air pollutants such as Carbon Monoxide (CO), Benzene (C6H6), Nitric Oxide (NO) and Nitrogen Dioxide (NO_2) and weather parameters like temperature, relative humidity and absolute humidity. Understanding how these factors interact over time allows for more accurate evaluations of air quality dynamics in urban contexts.

c) **Providing data-driven insights**

Provide actionable insights that can help with the development and installation of smart air quality monitoring systems. These technologies can help municipal authorities and urban planners in making proactive choices, such as providing pollution alerts, optimizing traffic flow for reduced emissions, or changing industrial activities depending on environmental criteria.

d) **Create the groundwork for future applications**

Create the groundwork for future applications, such as machine learning models for predicting air quality and detecting anomalies in real time. These advanced technologies may be linked into larger smart city infrastructures, helping to support automatic response mechanisms, citizen health advisory systems, and long-term environmental planning.

Through this analysis, this project aims to demonstrate how high-frequency environmental sensor data informs the development of intelligent urban systems. By integrating environmental data into smart city decision-making processes, municipalities may develop healthier, more resilient, and adaptable urban living environments in which public health and environmental quality are constantly monitored and actively maintained.

3.0 Literature Review

Integrating environmental monitoring technologies into municipal infrastructure has become a critical component of current smart city development. Air quality, in particular, has attracted a great deal of attention because of its direct influence on public health and environmental sustainability. Numerous studies have been conducted to address the complicated difficulties provided by urban air pollution.

I. The Emergence of Affordable Sensing Technologies for Urban Air Pollution Management

According to Kumar et al. (2015), smart cities leverage sensor networks and IoT (Internet of Things) technologies to collect real-time environmental data, enabling proactive responses to air quality issues. Urban environments are often exposed to harmful pollutants such as CO, NO_2, and PM2.5, and monitoring these in real time is crucial for public health and regulatory compliance. Smart cities such as Barcelona have already implemented such monitoring systems to guide transportation policies and issue public health alerts. Benavides et al. (2021) describe how Barcelona has implemented various traffic restriction initiatives aimed at decreasing the number of vehicles on the road to enhance air quality. These initiatives include the development of 'superblocks' to minimize space for private vehicles and the introduction of a city-wide Low Emission Zone (LEZ) that limits access for the most polluting vehicles. A recent study quantifies the effects of these initiatives, showing that when paired with fleet upgrades and reduced demand, significant global reductions in NOx emissions can be achieved, along with considerable drops in NO2 levels at traffic monitoring stations. This data-driven approach not only enhances urban resilience but also supports public health by informing transportation policies and improving air quality.

II. Populations That May Be Affected by Traffic-Related Air Pollution in Seven Global Cities

Traffic-related air pollution (TRAP) poses significant health risks to populations in urban areas across the globe. Studies have estimated exposure levels in various cities, highlighting the urgent need for public health interventions. Lipsitt et al. (2012) conducted a comprehensive analysis that identified populations potentially exposed to high levels of TRAP in seven major cities: Beijing, New Delhi, Los Angeles, Toronto, Paris, and Barcelona. Utilizing geographic information science techniques, their research assessed the risk of exposure in both developed and developing countries, revealing varying levels of TRAP based on geographic and demographic factors. In urban settings, high population density and traffic congestion significantly increase exposure to pollutants like nitrogen oxides (NOx), as noted by Schiavon et al. (2016).

While Schiavon's study emphasizes the contribution of road traffic to atmospheric pollution, particularly nitrogen oxides, it does not specifically address the populations exposed to TRAP in the same seven cities analyzed by Lipsitt et al. This gap underscores the importance of targeted public health interventions in urban areas where traffic congestion exacerbates air quality issues, necessitating further research to understand the full impact of TRAP on vulnerable populations.

III. Field Calibration of a Group of Affordable, Commercially Available Sensors for Air Quality Monitoring: Part B - NO, CO, and CO2

The field calibration of low-cost sensors significantly enhances the accuracy of air quality monitoring for pollutants such as nitrogen monoxide (NO), carbon monoxide (CO), and carbon dioxide (CO2). Calibration is crucial due to the inherent non-linear responses of these sensors to various air pollutants and their susceptibility to environmental factors. Effective calibration methods can mitigate these issues, leading to more reliable data collection. Apostolopoulos et al. (2023) highlight that calibration reduces measurement errors, demonstrating an improvement in the mean error for NO2 from 9.4 ppb to 3 ppb through the application of machine learning models. Furthermore, Han et al. (2024) emphasize the importance of adapting calibration techniques to diverse climatic conditions, as their research on calibrating low-cost sensors for NO2, NO, CO, and ozone (O3) shows that factors such as calibration period length, pollutant concentration range, and time averaging are crucial for enhancing sensor performance in various urban environments.

Calibration techniques, particularly machine learning approaches like Random Forest and Long Short-Term Memory (LSTM) networks, have shown superior performance in correcting sensor outputs, significantly enhancing accuracy (Apostolopoulos et al., 2023). Additionally, dynamic baseline tracking methods isolate concentration signals from temperature and humidity effects, further improving sensor reliability (Han et al., 2024). Spinelle (2017) reinforces the significance of field calibration, noting that supervised learning techniques yield the best agreement with reference measurements, achieving the Data Quality Objective of 25% uncertainty set by the European Air Quality Directive. While Apostolopoulos et al. focus on the calibration of low-cost sensors for NO2 and O3, they do not specifically address the calibration impact on NO, CO, and CO2 monitoring accuracy, indicating a gap in the literature that warrants further investigation.

IV. The Changing Paradigm of Air Pollution Monitoring

The changing paradigm of air pollution monitoring is significantly influenced by the integration of machine learning (ML) and artificial intelligence (AI), which enhances the accuracy and efficiency of monitoring systems through advanced algorithms and real-time data processing. These technologies facilitate the development of sophisticated models capable of predicting air quality levels, identifying pollution sources, and providing timely alerts, ultimately improving public health outcomes. Aghazada (2024) highlights that various ML techniques, such as decision trees and deep learning models like AQNet-CNN, are employed to analyze air quality data, leading to improved prediction accuracy for pollutants such as PM2.5 and NO2. Similarly, Rajak et al. (2024) emphasize the role of hybrid models in interpreting sensor data, which enhances forecasting accuracy and reliability for these pollutants by utilizing performance evaluation metrics to refine predictive capabilities.

Moreover, the integration of Internet of Things (IoT) devices with AI allows for real-time data collection and analysis, enabling immediate responses to pollution events (Banciu et al., 2024). This combination not only enhances the responsiveness of monitoring systems but also facilitates decentralized systems where edge AI implementations reduce latency and bandwidth usage, improving overall system efficiency (Ramu et al., 2024). Continuous learning and adaptation are also key features of these AI systems, as they can self-improve by updating their models based on incoming data, thereby enhancing their predictive capabilities over time (Kumar, 2024; Ramu et al., 2024).

While the integration of AI and ML in air pollution monitoring shows great promise, challenges remain, such as the need for large-scale trials and potential data privacy concerns in decentralized systems. Addressing these issues will be crucial for the widespread adoption of these technologies in environmental management, ensuring that they can effectively contribute to better air quality and public health.

V. Amperometric Gas Sensors as an Affordable, Emerging Technological Platform for Air Quality Monitoring Applications

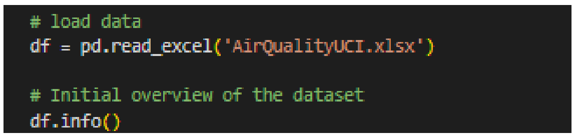
Amperometric gas sensors are emerging as a promising and affordable technology platform for air quality monitoring, particularly effective in detecting inorganic gases that have a significant impact on urban air quality, including carbon monoxide (CO), nitrogen dioxide (NO2), ozone (O3), and sulfur dioxide (SO2). The development of these sensors has been fueled by the growing demand for devices that are sensitive, selective, and stable, allowing for deployment in dense networks to provide real-time air quality data. This capability is essential for overcoming the limitations of traditional air quality monitoring stations, which are often limited in number and have longer recording intervals (Baron & Saffell, 2017).

Amperometric sensors offer several key advantages for urban air quality monitoring applications. Their cost-effectiveness stems from printed electronics manufacturing strategies that enable scalable, low-cost production, making widespread deployment in urban environments economically feasible without prohibitive expenses (Carter et al., 2012). These sensors deliver high performance through excellent sensitivity and selectivity, with low detection limits and long durability that outperform other low-cost alternatives like heated metal oxide semiconductor platforms, effectively bridging the gap between affordability and performance (Carter et al., 2012). This combination of affordability, performance, and energy efficiency makes amperometric sensors particularly valuable for accurately measuring toxic gases in urban settings where air quality can fluctuate rapidly.

4.0 Methodology

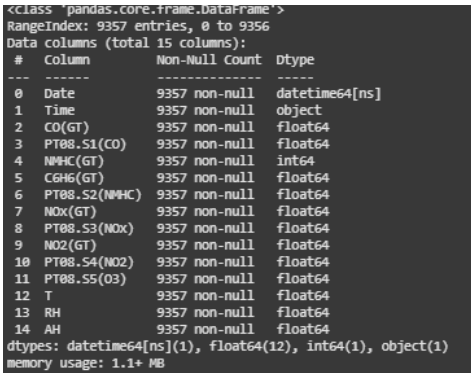
I. Dataset Acquisition and Size

For this predictive modelling endeavour, a widely recognised and publicly available dataset, the "Air Quality UCI" dataset, was meticulously selected. This particular dataset is held in high regard within environmental science and machine learning communities, largely due to its comprehensive nature and its ability to capture a critical aspect of urban air quality. It contains hourly averaged responses derived from an array of five metal oxide chemical sensors, which are meticulously embedded within a sophisticated air quality chemical multi-sensor device. Crucially, these sensor readings are complemented by highly accurate reference values for key atmospheric pollutants, including Carbon Monoxide (CO), non-methane hydrocarbons (NMHC), Benzene, Nitrogen Oxides (NOx), and Nitrogen Dioxide (NO2). The data itself was collected over a significant period within a heavily polluted road environment, specifically located in Rome, Italy, thereby providing a robust foundation for analysing real-world air quality dynamics. Following the initial process of data acquisition and loading into the analytical environment, the dataset was found to comprise 9357 distinct entries, distributed across 15 informative columns. This was formally confirmed by the df.info() output upon initial inspection, and further exemplified by the dataset's initial shape and first few rows shown in Figure 4.1.2. This dataset's comprehensive nature and larger dimensions allow for more robust model training and potentially better generalisation capabilities than a smaller dataset might otherwise offer.



```
# load data
df = pd.read_excel('AirQualityUCI.xlsx')

# Initial overview of the dataset
df.info()
```

"Figure 4.1.1: Initial Dataset Information."



```
<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   datetime64[ns]
 1   Time           9357 non-null   object
 2   CO(GT)         9357 non-null   float64
 3   PT08.S1(CO)    9357 non-null   float64
 4   NMHC(GT)       9357 non-null   int64
 5   C6H6(GT)       9357 non-null   float64
 6   PT08.S2(NMHC)  9357 non-null   float64
 7   NOx(GT)        9357 non-null   float64
 8   PT08.S3(NOx)   9357 non-null   float64
 9   NO2(GT)        9357 non-null   float64
 10  PT08.S4(NO2)   9357 non-null   float64
 11  PT08.S5(O3)    9357 non-null   float64
 12  T              9357 non-null   float64
 13  RH             9357 non-null   float64
 14  AH             9357 non-null   float64
dtypes: datetime64[ns](1), float64(12), int64(1), object(1)
memory usage: 1.1+ MB
```

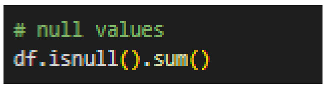
"Figure 4.1.2: Initial Dataset Information."

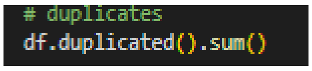
II. Data Cleaning and Exploratory Data Analysis (EDA)

Upon the initial, thorough inspection of the acquired dataset, it was genuinely gratifying to find it in such a remarkably clean state, a solid foundation for any predictive modelling endeavour. A comprehensive and systematic check for any missing values was diligently carried out using the df.isnull().sum() function. This meticulous examination revealed a complete absence of gaps across every single column, an ideal starting point that immediately streamlined the data preparation process. Concurrently, an equally rigorous examination for duplicate rows, performed through df.duplicated().sum(), unequivocally confirmed that there were zero redundant entries within the dataset, as clearly shown in Figure 4.2.2. This intrinsic high quality of the raw data proved to be a significant advantage as it fundamentally removed the need for extensive and often time-consuming imputation techniques or the removal of numerous rows that might otherwise arise from widespread missing or repetitive information.

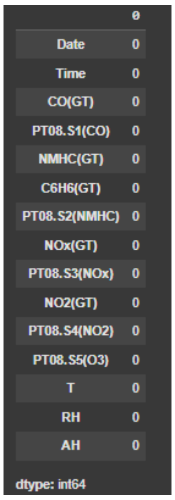
Additionally, the cleaning procedure included carefully dealing with data problems that are common with real-world sensor data, especially when it comes to outliers. The outlier treatment plan was meticulously tailored, taking into account the specific statistical characteristics and environmental context of each particular feature, rather than using a one-size-fits-all approach. For instance, outliers identified in CO(GT), NO2(GT), and NOx(GT) were deliberately 'kept' as these extreme values often genuinely represent critical pollution events or unusual but valid atmospheric conditions, rather than erroneous readings. In contrast, for sensor responses such as PT08.S1(CO), PT08.S2(NMHC), PT08.S3(NOx), PT08.S4(NO2), and PT08.S5(O3), alongside 'AH' (Absolute Humidity), these were capped at their 99th percentile. This approach acknowledged that physical sensors can saturate or produce abnormally high readings at extreme concentrations. Conversely, NMHC(GT) and the target variable C6H6(GT) were capped at the 95th percentile; this decision aimed at mitigating the undue influence of extreme spikes while still crucially retaining information about significant, albeit less frequent, pollution events. Meteorological features were also handled specifically: Temperature ('T') values were meticulously filtered to ensure they fell within a physically valid range of -20°C to 50°C, and Relative Humidity ('RH') was 'clipped' to its physically impossible range of 0-100%. Finally, any derived features, such as 'Temp_Hum_Interaction', were recalculated after their component treatments to ensure consistency. This multi-faceted and targeted approach significantly reduced the number of outliers, with notable improvements observed across several key features. For example, PT08.S1(CO) outliers were reduced from 484 to 117, as can be seen in Figure 4.2.8, thereby substantially enhancing the overall data quality for subsequent modelling efforts. This process is evidenced by the detailed outlier treatment logs in Figure 4.2.6 and further illuminated by the comprehensive summary and comparative visualisations in Figure 4.2.8.

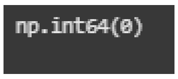
For the crucial phase of Exploratory Data Analysis (EDA), the distribution of Carbon Monoxide (CO(GT)) concentration was visualised using a histogram complemented by a Kernel Density Estimate (KDE). This visualisation proved instrumental in revealing its overall distribution patterns, offering valuable initial insights into the data. The plot clearly illustrated a noticeable right-skew in the CO(GT) concentrations, indicating that lower concentration values were far more frequent, with higher concentrations occurring less often. This kind of distribution is common in environmental data and suggests that while air quality is often good, occasional pollution events lead to elevated readings. This initial exploration helped in understanding the typical range and variability of a key pollutant, informing subsequent modelling considerations, as shown in Figure 4.2.4.


```
# null values
df.isnull().sum()
```

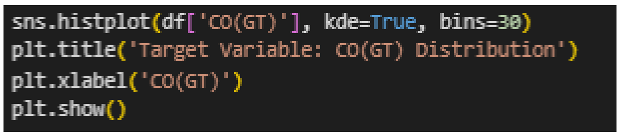

```
# duplicates
df.duplicated().sum()
```

"Figure 4.2.1: Data Cleanliness Verification."


```
                0
         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   0
dtype: int64
```

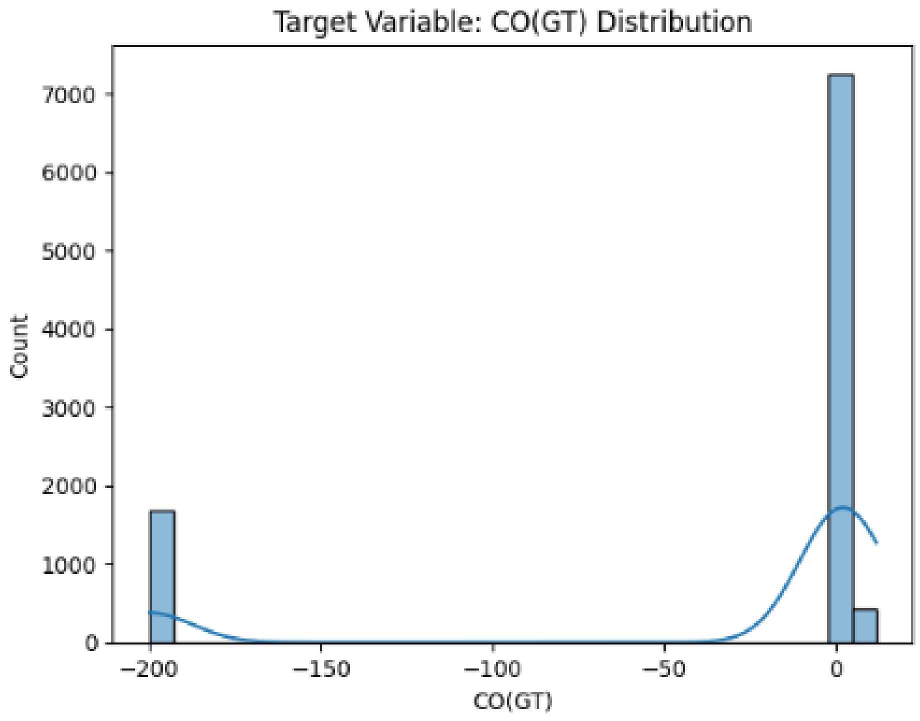

```
np.int64(0)
```

"Figure 4.2.2: Output Data Cleanliness Verification."

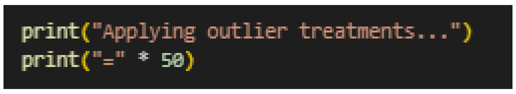


```
sns.histplot(df['CO(GT)'], kde=True, bins=30)
plt.title('Target Variable: CO(GT) Distribution')
plt.xlabel('CO(GT)')
plt.show()
```

"Figure 4.2.3: Distribution of CO(GT) Concentration."

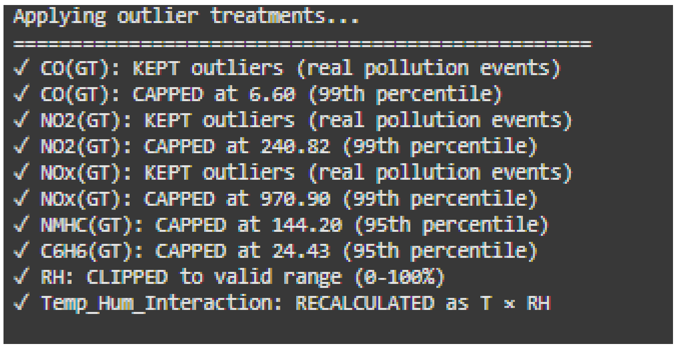


"Figure 4.2.4: Output Distribution of CO(GT) Concentration."



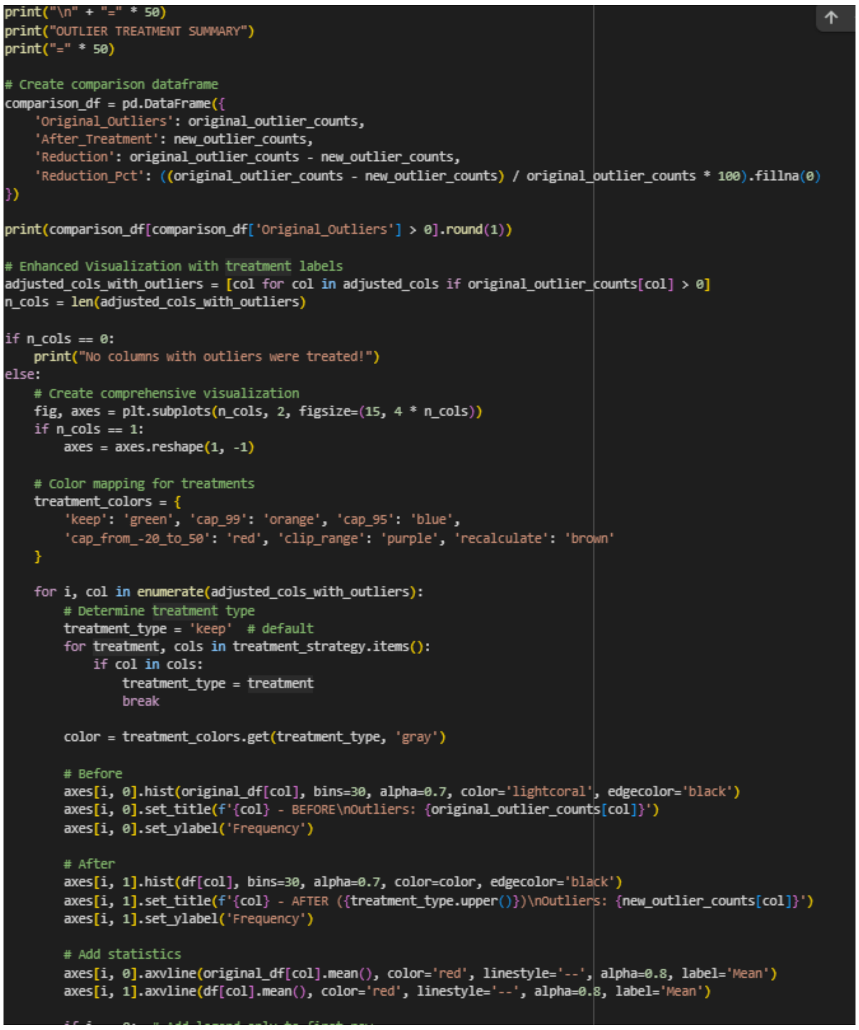
```
print("Applying outlier treatments...")
print("=" * 50)
```

"Figure 4.2.5: Outlier Treatment Application Logs."

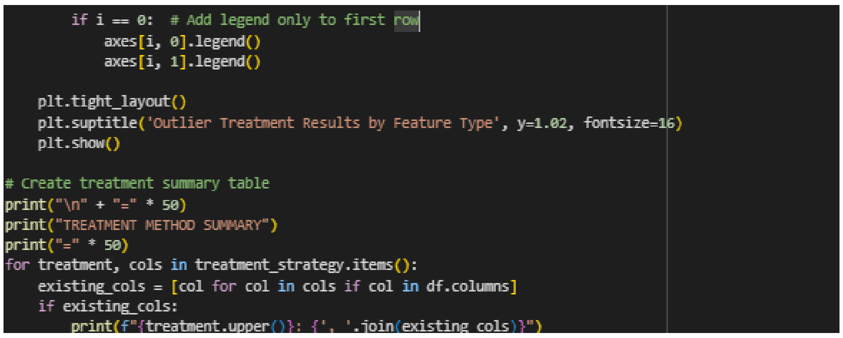


```
Applying outlier treatments...
==================================================
✓ CO(GT): KEPT outliers (real pollution events)
✓ CO(GT): CAPPED at 6.60 (99th percentile)
✓ NO2(GT): KEPT outliers (real pollution events)
✓ NO2(GT): CAPPED at 240.82 (99th percentile)
✓ NOx(GT): KEPT outliers (real pollution events)
✓ NOx(GT): CAPPED at 970.90 (99th percentile)
✓ NMHC(GT): CAPPED at 144.20 (95th percentile)
✓ C6H6(GT): CAPPED at 24.43 (95th percentile)
✓ RH: CLIPPED to valid range (0-100%)
✓ Temp_Hum_Interaction: RECALCULATED as T × RH
```

"Figure 4.2.6: Output Outlier Treatment Application Logs."

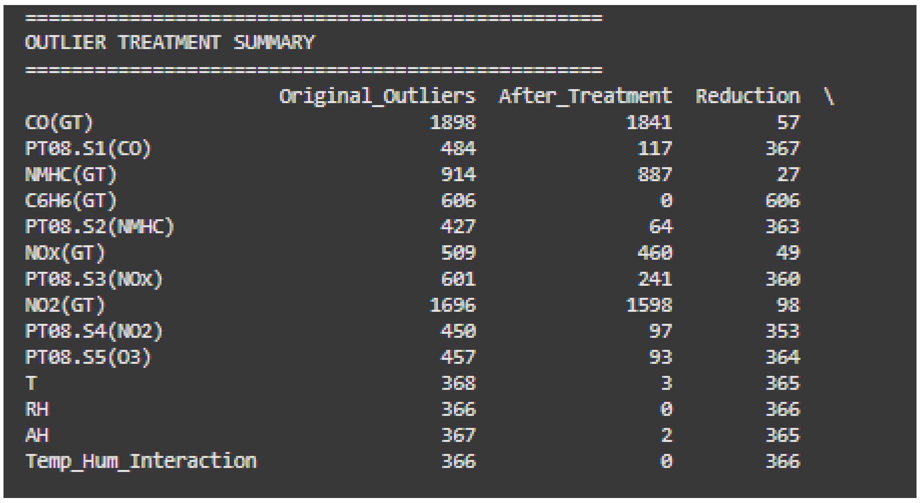


```python
print("\n" + "=" * 50)
print("OUTLIER TREATMENT SUMMARY")
print("=" * 50)

# Create comparison dataframe
comparison_df = pd.DataFrame({
    'Original_Outliers': original_outlier_counts,
    'After_Treatment': new_outlier_counts,
    'Reduction': original_outlier_counts - new_outlier_counts,
    'Reduction_Pct': ((original_outlier_counts - new_outlier_counts) / original_outlier_counts * 100).fillna(0)
})

print(comparison_df[comparison_df['Original_Outliers'] > 0].round(1))

# Enhanced Visualization with treatment labels
adjusted_cols_with_outliers = [col for col in adjusted_cols if original_outlier_counts[col] > 0]
n_cols = len(adjusted_cols_with_outliers)

if n_cols == 0:
    print("No columns with outliers were treated!")
else:
    # Create comprehensive visualization
    fig, axes = plt.subplots(n_cols, 2, figsize=(15, 4 * n_cols))
    if n_cols == 1:
        axes = axes.reshape(1, -1)

    # Color mapping for treatments
    treatment_colors = {
        'keep': 'green', 'cap_99': 'orange', 'cap_95': 'blue',
        'cap_from_-20_to_50': 'red', 'clip_range': 'purple', 'recalculate': 'brown'
    }

    for i, col in enumerate(adjusted_cols_with_outliers):
        # Determine treatment type
        treatment_type = 'keep'  # default
        for treatment, cols in treatment_strategy.items():
            if col in cols:
                treatment_type = treatment
                break

        color = treatment_colors.get(treatment_type, 'gray')

        # Before
        axes[i, 0].hist(original_df[col], bins=30, alpha=0.7, color='lightcoral', edgecolor='black')
        axes[i, 0].set_title(f'{col} - BEFORE\nOutliers: {original_outlier_counts[col]}')
        axes[i, 0].set_ylabel('Frequency')

        # After
        axes[i, 1].hist(df[col], bins=30, alpha=0.7, color=color, edgecolor='black')
        axes[i, 1].set_title(f'{col} - AFTER ({treatment_type.upper()})\nOutliers: {new_outlier_counts[col]}')
        axes[i, 1].set_ylabel('Frequency')

        # Add statistics
        axes[i, 0].axvline(original_df[col].mean(), color='red', linestyle='--', alpha=0.8, label='Mean')
        axes[i, 1].axvline(df[col].mean(), color='red', linestyle='--', alpha=0.8, label='Mean')
```

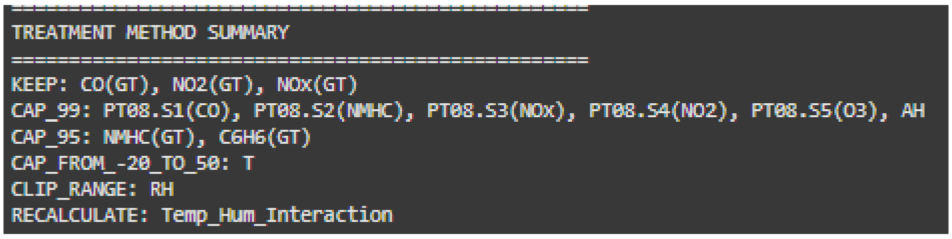


```python
        if i == 0:  # Add legend only to first row
            axes[i, 0].legend()
            axes[i, 1].legend()

    plt.tight_layout()
    plt.suptitle('Outlier Treatment Results by Feature Type', y=1.02, fontsize=16)
    plt.show()

# Create treatment summary table
print("\n" + "=" * 50)
print("TREATMENT METHOD SUMMARY")
print("=" * 50)
for treatment, cols in treatment_strategy.items():
    existing_cols = [col for col in cols if col in df.columns]
    if existing_cols:
        print(f"{treatment.upper()}: {', '.join(existing_cols)}")
```

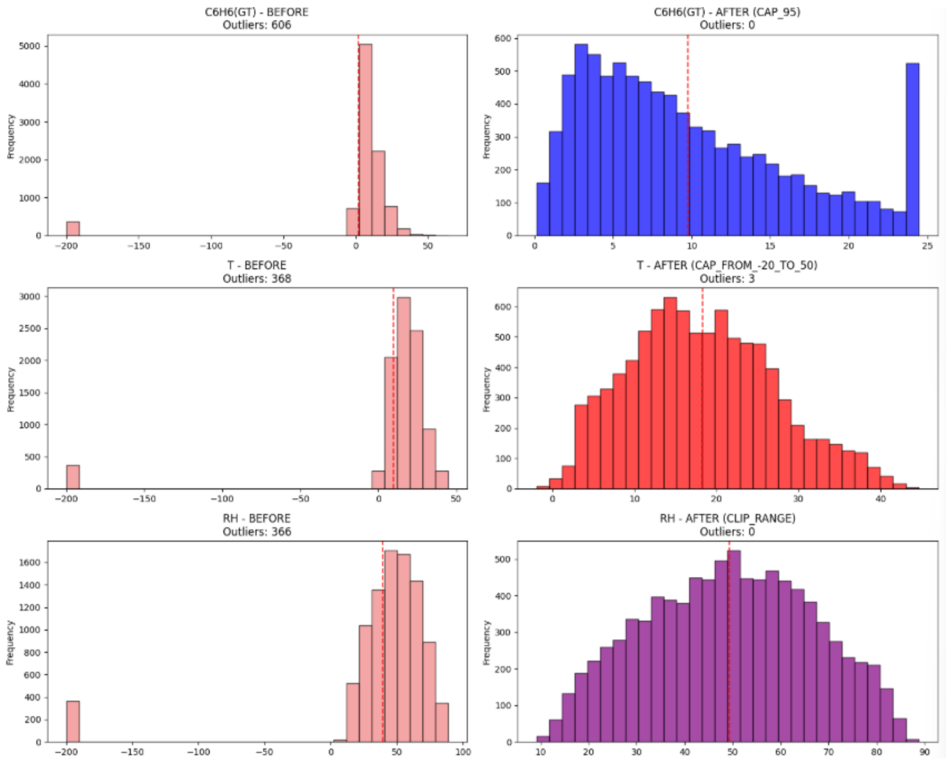
"Figure 4.2.7: Outlier Treatment Summary and Visualisation."



```
==================================================
OUTLIER TREATMENT SUMMARY
==================================================
                      Original_Outliers  After_Treatment  Reduction  \
CO(GT)                             1898             1841         57
PT08.S1(CO)                         484              117        367
NMHC(GT)                            914              887         27
C6H6(GT)                            606                0        606
PT08.S2(NMHC)                       427               64        363
NOx(GT)                             509              460         49
PT08.S3(NOx)                        601              241        360
NO2(GT)                            1696             1598         98
PT08.S4(NO2)                        450               97        353
PT08.S5(O3)                         457               93        364
T                                   368                3        365
RH                                  366                0        366
AH                                  367                2        365
Temp_Hum_Interaction                366                0        366
```



```
==================================================
TREATMENT METHOD SUMMARY
==================================================
KEEP: CO(GT), NO2(GT), NOx(GT)
CAP_99: PT08.S1(CO), PT08.S2(NMHC), PT08.S3(NOx), PT08.S4(NO2), PT08.S5(O3), AH
CAP_95: NMHC(GT), C6H6(GT)
CAP_FROM_-20_TO_50: T
CLIP_RANGE: RH
RECALCULATE: Temp_Hum_Interaction
```

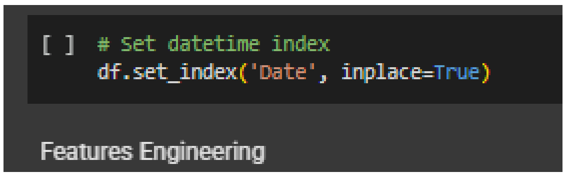


"Figure 4.2.8: Output Outlier Treatment Summary and Visualisation."

III. Feature Engineering and Number of Features

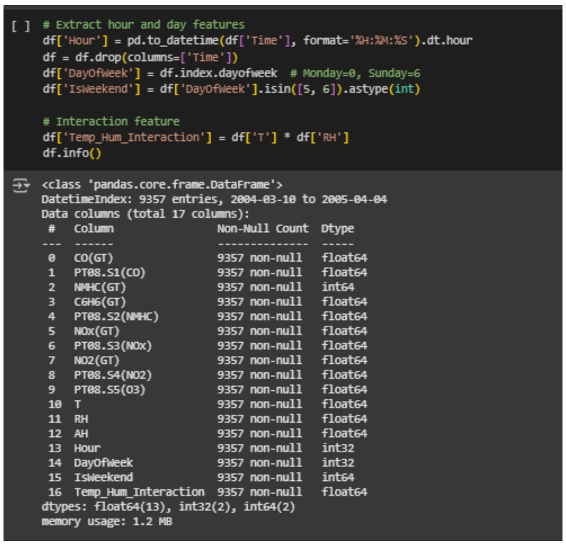
An important step in feature engineering was carried out to greatly improve the model's ability to predict and its clarity of understanding. This process included the removal of new, timely features directly from the current 'Date' and 'Time' columns, thus converting raw timestamps into significant characteristics. At first, the 'Date' column was designated as the index for the DataFrame. Afterward, the developed features comprised 'Hour', carefully taken from the 'Time' column. It is essential to recognize the hourly patterns that occur regularly in air quality changes. For instance, higher pollution levels often align with busy traffic times. The 'DayOfWeek' extracted from the DataFrame's index offered valuable information regarding weekly patterns and changes that may occur due to different human behaviors on weekdays compared to weekends. Additionally, 'IsWeekend' was defined as a binary feature to signify whether a given day fell on a weekend. This allows the model to take into consideration the different air quality patterns that are seen on days when people are not working. A new feature called 'Temp_Hum_Interaction' was also developed by multiplying 'T' (Temperature) and 'RH' (Relative Humidity), designed to record the combined impacts of the environment.

After the successful extraction and incorporation of these new features, the original 'Time' column was carefully eliminated from the dataset. The essential information has been thoroughly gathered and represented by the newly created time-related features, thereby eliminating redundancy and minimizing complexity. The information in the updated DataFrame, which now includes these new columns, confirmed that the feature engineering was successfully carried out, as shown in Figure 4.3.1. The final collection of attributes chosen for training the predictive model consists of 15 unique variables, which include the newly created ones. This extensive set offers a valuable range of information for the learning algorithm. A correlation heatmap was created to visually evaluate the connections among these features, which helps in understanding possible multicollinearity, as shown in Figure 4.3.2.



```
[ ]  # Set datetime index
     df.set_index('Date', inplace=True)
```

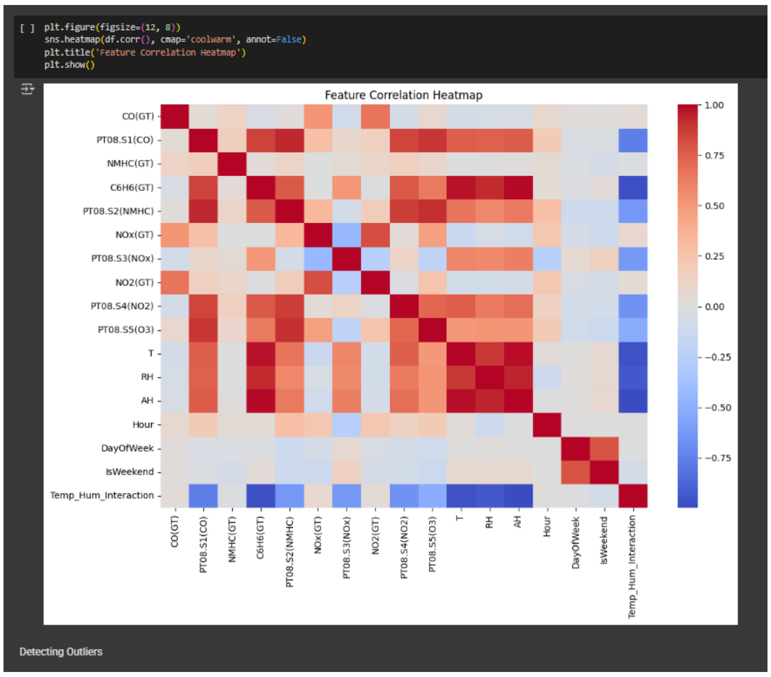
Features Engineering



```
[ ]  # Extract hour and day features
     df['Hour'] = pd.to_datetime(df['Time'], format='%H:%M:%S').dt.hour
     df = df.drop(columns=['Time'])
     df['DayOfWeek'] = df.index.dayofweek  # Monday=0, Sunday=6
     df['IsWeekend'] = df['DayOfWeek'].isin([5, 6]).astype(int)

     # Interaction feature
     df['Temp_Hum_Interaction'] = df['T'] * df['RH']
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 9357 entries, 2004-03-10 to 2005-04-04
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CO(GT)                9357 non-null   float64
 1   PT08.S1(CO)           9357 non-null   float64
 2   NMHC(GT)              9357 non-null   int64
 3   C6H6(GT)              9357 non-null   float64
 4   PT08.S2(NMHC)         9357 non-null   float64
 5   NOx(GT)               9357 non-null   float64
 6   PT08.S3(NOx)          9357 non-null   float64
 7   NO2(GT)               9357 non-null   float64
 8   PT08.S4(NO2)          9357 non-null   float64
 9   PT08.S5(O3)           9357 non-null   float64
 10  T                     9357 non-null   float64
 11  RH                    9357 non-null   float64
 12  AH                    9357 non-null   float64
 13  Hour                  9357 non-null   int32
 14  DayOfWeek             9357 non-null   int32
 15  IsWeekend             9357 non-null   int64
 16  Temp_Hum_Interaction  9357 non-null   float64
dtypes: float64(13), int32(2), int64(2)
memory usage: 1.2 MB
```

"Figure 4.3.1: Feature Engineering and Updated DataFrame Information."



```
[ ]  plt.figure(figsize=(12, 8))
     sns.heatmap(df.corr(), cmap='coolwarm', annot=False)
     plt.title('Feature Correlation Heatmap')
     plt.show()
```

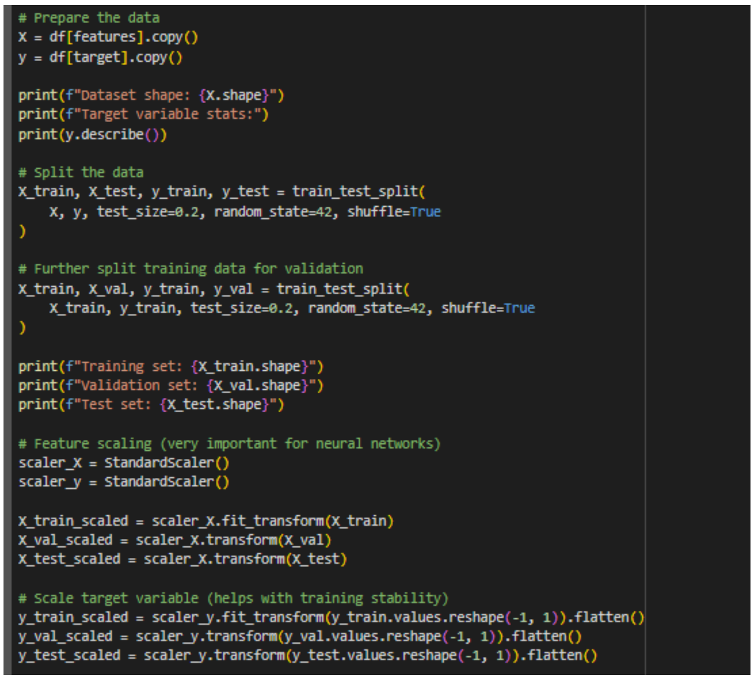
Detecting Outliers

"Figure 4.3.2: Feature Correlation Heatmap."

IV. Model Selection (Artificial Neural Networks) and Evaluation Metrics

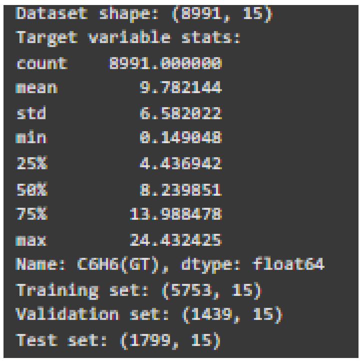
For the central task of predicting air quality, an Artificial Neural Network (ANN) was deliberately chosen as the primary machine learning algorithm. ANNs demonstrate particular suitability for this domain, owing to their inherent capacity to discern and model the highly complex, non-linear relationships frequently observed within environmental time-series data, even without being explicitly structured as recurrent networks. The ability of ANNs to learn intricate patterns from diverse input features makes them an ideal choice for tasks where traditional linear models might prove insufficient. Given that the target variable, Benzene concentration (C6H6(GT)), is fundamentally a continuous numerical value, such as measured in μg/m3, the problem was unequivocally framed as a regression task. This approach aimed to predict a precise numerical output rather than assigning a categorical class.

The basic characteristics of this regression issue require the application of assessment measures that are specifically created for continuous numerical forecasts. The model's performance was thoroughly assessed using recognized and appropriate regression metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the R^2 Score (Coefficient of Determination). The dataset was carefully split into training, validation, and test sets to ensure a comprehensive evaluation. Specifically, the dataset for modelling comprised 8991 samples after outlier filtering. From this, the training set comprised 5753 samples, the validation set 1439 samples, and the test set 1799 samples, reflecting approximate split ratios of 64%, 16%, and 20%, respectively. The target variable (C6H6(GT)) exhibited a mean of approximately 9.782 and a standard deviation of 6.582, indicating its distribution range. All features and the target variable were then scaled using StandardScaler, a crucial step for optimal neural network training, as illustrated by the dataset shapes in Figure 4.4.1. This ensured that features contributed equally to the model training process. It is thus acknowledged that while classification-based evaluation metrics might be specified elsewhere, the current implementation aligns with a regression model, and therefore, appropriate regression metrics were employed.



```python
# Prepare the data
X = df[features].copy()
y = df[target].copy()

print(f"Dataset shape: {X.shape}")
print(f"Target variable stats:")
print(y.describe())

# Split the data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, shuffle=True
)

# Further split training data for validation
X_train, X_val, y_train, y_val = train_test_split(
    X_train, y_train, test_size=0.2, random_state=42, shuffle=True
)

print(f"Training set: {X_train.shape}")
print(f"Validation set: {X_val.shape}")
print(f"Test set: {X_test.shape}")

# Feature scaling (very important for neural networks)
scaler_X = StandardScaler()
scaler_y = StandardScaler()

X_train_scaled = scaler_X.fit_transform(X_train)
X_val_scaled = scaler_X.transform(X_val)
X_test_scaled = scaler_X.transform(X_test)

# Scale target variable (helps with training stability)
y_train_scaled = scaler_y.fit_transform(y_train.values.reshape(-1, 1)).flatten()
y_val_scaled = scaler_y.transform(y_val.values.reshape(-1, 1)).flatten()
y_test_scaled = scaler_y.transform(y_test.values.reshape(-1, 1)).flatten()
```

"Figure 4.4.1: Data Preparation and Splitting."



```
Dataset shape: (8991, 15)
Target variable stats:
count    8991.000000
mean        9.782144
std         6.582022
min         0.149048
25%         4.436942
50%         8.239851
75%        13.988478
max        24.432425
Name: C6H6(GT), dtype: float64
Training set: (5753, 15)
Validation set: (1439, 15)
Test set: (1799, 15)
```

"Figure 4.4.2: Output Data Preparation and Splitting."

5.0 ANN Architecture

The task of estimating Benzene concentration was undertaken by using the functionalities of Artificial Neural Networks (ANNs), particularly through an advanced Multi-Layer Perceptron (MLP). This design decision was made possible by the advanced Keras API, which is paired with a TensorFlow backend, collectively offering a flexible and strong framework for building the model. A step-by-step structure was utilized, enabling data to move methodically through a set of linked layers. Central to this design was the careful use of sophisticated methods like Batch Normalization and Dropout layers. These enhancements were essential for improving the network's training stability, speeding up its progress toward finding the best solution, and offering strong regularization to successfully avoid overfitting. This approach guarantees that the model can identify true patterns instead of just irrelevant noise. The complete framework of the network was contained within a reusable Python function, create_mlp_model, allowing for easy testing and improvement of hidden layer settings, dropout rates, and learning parameters.

I. Network Architecture Details

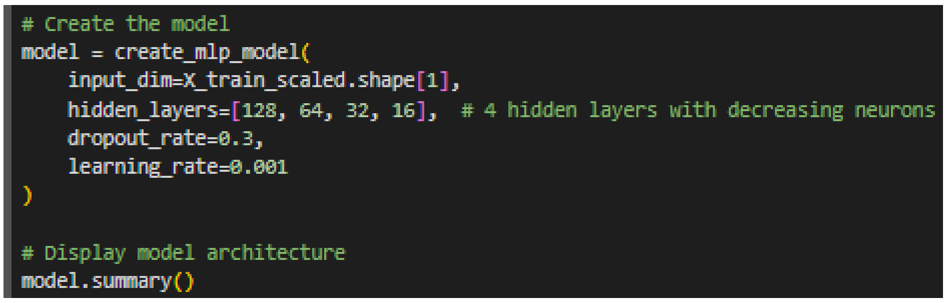
The model's predictive capability is fundamentally rooted in its carefully designed architecture. It begins by implicitly accepting the **15 features** that were carefully engineered in the preceding data preparation phase. These features then progress through a sequence of four dense hidden layers; each purposefully designed to progressively extract and learn increasingly abstract and complex representations of the input data:

- **Hidden Layer 1:** This initial processing layer is configured with **128 units**. For activation, the **Rectified Linear Unit (ReLU)** was chosen, a popular non-linear function in deep learning, valued for its computational efficiency and its effectiveness in mitigating issues like vanishing gradients, which can impede learning in deeper networks. Immediately following this dense layer, a **Batch Normalisation** layer was introduced. This vital component standardises the inputs to the next layer, a technique that significantly stabilises the learning process and allows for the utilisation of higher learning rates, ultimately accelerating training. The implementation of a 30% Dropout layer (p=0.3) selectively disables neural units during training to enhance generalization capability. This intelligent mechanism compels the network to develop more robust features, preventing it from over-relying on any single input connection, thereby encouraging the learning of more generalised patterns.

- **Hidden Layer 2:** Building upon the rich representations acquired by the first layer, this layer comprises **64 units**, also activated by **ReLU**. Consistent with the established regularisation strategy, it is immediately succeeded by both a Batch Normalisation layer and another Dropout layer (also with a 0.3 rate), further enhancing the model's ability to generalise effectively to unseen data.
- **Hidden Layer 3:** Progressively refining the feature space, this layer consists of **32 units** with **ReLU** activation, followed by yet another pair of Batch Normalisation and Dropout (0.3 rate) layers. This systematic reduction in units assists the network in distilling the most pertinent information and extracting hierarchical features more effectively.
- **Hidden Layer 4:** The final hidden layer in the sequence contains **16 units**, again employing **ReLU** activation, Batch Normalisation, and a 0.3 Dropout layer. This last stage of transformation ensures that the features presented to the output layer are highly refined and optimally relevant for the prediction task.

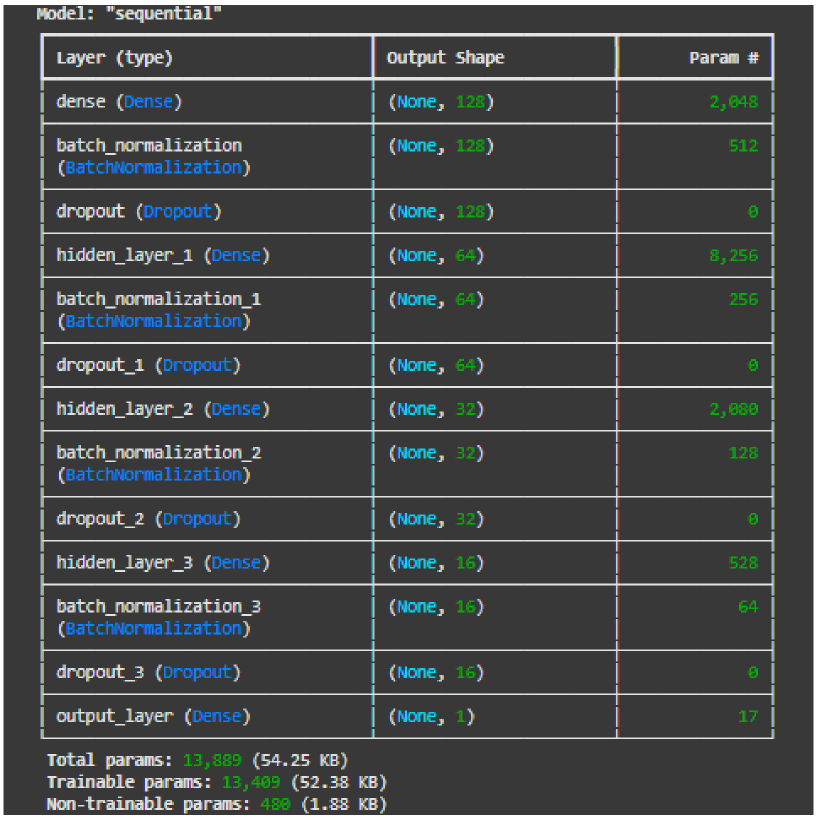
- **Output Layer:** The network culminates in a single **dense unit** responsible for generating the final prediction. Critically, a **linear activation function** was selected for this output layer. This choice is paramount for regression tasks, as it enables the model to directly output raw, unscaled, continuous numerical values. These values are subsequently inverse-transformed to represent the predicted Benzene concentration in its original, interpretable scale.

A comprehensive model.summary() output, as presented in Figure 5.1.2, provides a detailed breakdown of this architecture, confirming the configuration of each layer, their respective output shapes, and the parameter counts. It shows a **total of 13,889 parameters**, of which **13,409 were trainable**, signifying a well-considered model complexity that aligns effectively with the nature and size of the dataset and the learning task.



```python
# Create the model
model = create_mlp_model(
    input_dim=X_train_scaled.shape[1],
    hidden_layers=[128, 64, 32, 16],  # 4 hidden layers with decreasing neurons
    dropout_rate=0.3,
    learning_rate=0.001
)

# Display model architecture
model.summary()
```

"Figure 5.1.1: Artificial Neural Network Architecture Summary."



```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ dense (Dense)                   │ (None, 128)            │         2,048 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization             │ (None, 128)            │           512 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ hidden_layer_1 (Dense)          │ (None, 64)             │         8,256 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_1           │ (None, 64)             │           256 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_1 (Dropout)             │ (None, 64)             │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ hidden_layer_2 (Dense)          │ (None, 32)             │         2,080 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_2           │ (None, 32)             │           128 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_2 (Dropout)             │ (None, 32)             │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ hidden_layer_3 (Dense)          │ (None, 16)             │           528 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_3           │ (None, 16)             │            64 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_3 (Dropout)             │ (None, 16)             │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ output_layer (Dense)            │ (None, 1)              │            17 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 13,889 (54.25 KB)
 Trainable params: 13,409 (52.38 KB)
 Non-trainable params: 480 (1.88 KB)
```

"Figure 5.1.2: Output Artificial Neural Network Architecture Summary."

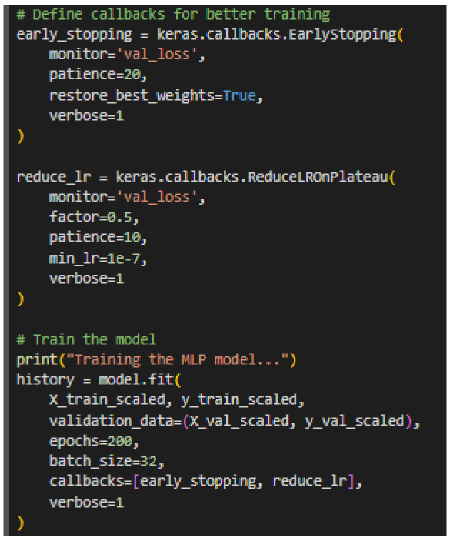
II. Model Compilation and Training

To prepare the network for learning from the data, the model was meticulously compiled. The **Adam optimiser** was chosen for its efficiency and adaptability in handling sparse gradients and non-stationary objectives, configured with a precise **learning rate of 0.001**. As this is a regression problem, the **Mean Squared Error (MSE)** was designated as the primary loss function, aiming to minimise the average of the squared differences between predicted and actual Benzene concentrations. Additionally, Mean Absolute Error (MAE) was included as a metric to monitor during training, offering a more interpretable measure of the average error magnitude in the original units.

The training regimen involved iterating over the dataset for **200 epochs**, with a **batch_size of 32**. This configuration meant that the model's internal weights were updated after processing batches of 32 samples at a time, striking a balance between training speed and the stability of learning. To ensure robust training and to proactively guard against the perils of overfitting, two critical callbacks were implemented:

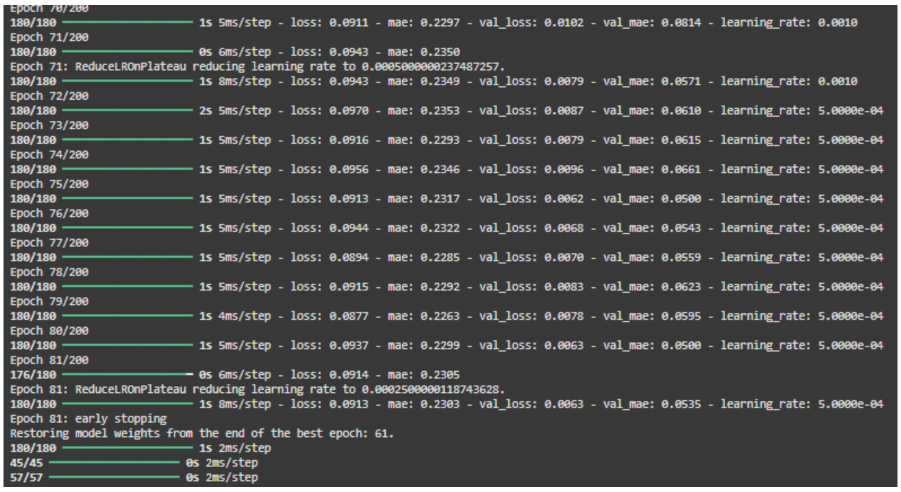
- **EarlyStopping:** This intelligent callback was configured to monitor the val_loss (validation loss). With a patience of 20 epochs, it would diligently observe the model's performance on the validation set. If the validation loss showed no improvement for 20 consecutive epochs, EarlyStopping would gracefully halt the training process, preventing wasted computational resources and automatically restoring the model weights that yielded the best validation performance.
- **ReduceLROnPlateau:** This callback was also set to monitor val_loss, with a factor of 0.5 and a patience of 10 epochs. Should the validation loss stagnate for 10 epochs, this mechanism would dynamically reduce our optimiser's learning rate by half. This strategy allows the model to make finer adjustments in its weight updates, potentially helping it to escape subtle local minima and continue converging more effectively. The minimum learning rate was constrained to 1e−7 to prevent excessively small updates.

The training process, meticulously monitored through detailed epoch-by-epoch logs (as vividly exemplified in Figure 5.2.2), consistently showcased a reduction in both training and validation loss, suggesting effective learning and stable convergence. This compelling evidence affirmed the efficacy of our chosen architecture and training parameters in capturing the intricate patterns of air quality data.



```
# Define callbacks for better training
early_stopping = keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=20,
    restore_best_weights=True,
    verbose=1
)

reduce_lr = keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=10,
    min_lr=1e-7,
    verbose=1
)

# Train the model
print("Training the MLP model...")
history = model.fit(
    X_train_scaled, y_train_scaled,
    validation_data=(X_val_scaled, y_val_scaled),
    epochs=200,
    batch_size=32,
    callbacks=[early_stopping, reduce_lr],
    verbose=1
)
```

"Figure 5.2.1: Model Training Progress (Sample Output)."



```
Epoch 70/200
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0911 - mae: 0.2297 - val_loss: 0.0102 - val_mae: 0.0814 - learning_rate: 0.0010
Epoch 71/200
180/180 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0943 - mae: 0.2350
Epoch 71: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0943 - mae: 0.2349 - val_loss: 0.0079 - val_mae: 0.0571 - learning_rate: 0.0010
Epoch 72/200
180/180 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 0.0970 - mae: 0.2353 - val_loss: 0.0087 - val_mae: 0.0610 - learning_rate: 5.0000e-04
Epoch 73/200
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0916 - mae: 0.2293 - val_loss: 0.0079 - val_mae: 0.0615 - learning_rate: 5.0000e-04
Epoch 74/200
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0956 - mae: 0.2346 - val_loss: 0.0096 - val_mae: 0.0661 - learning_rate: 5.0000e-04
Epoch 75/200
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0913 - mae: 0.2317 - val_loss: 0.0062 - val_mae: 0.0500 - learning_rate: 5.0000e-04
Epoch 76/200
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0944 - mae: 0.2322 - val_loss: 0.0068 - val_mae: 0.0543 - learning_rate: 5.0000e-04
Epoch 77/200
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0894 - mae: 0.2285 - val_loss: 0.0070 - val_mae: 0.0559 - learning_rate: 5.0000e-04
Epoch 78/200
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0915 - mae: 0.2292 - val_loss: 0.0083 - val_mae: 0.0623 - learning_rate: 5.0000e-04
Epoch 79/200
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0877 - mae: 0.2263 - val_loss: 0.0078 - val_mae: 0.0595 - learning_rate: 5.0000e-04
Epoch 80/200
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0937 - mae: 0.2299 - val_loss: 0.0063 - val_mae: 0.0500 - learning_rate: 5.0000e-04
Epoch 81/200
176/180 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0914 - mae: 0.2305
Epoch 81: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0913 - mae: 0.2303 - val_loss: 0.0063 - val_mae: 0.0535 - learning_rate: 5.0000e-04
Epoch 81: early stopping
Restoring model weights from the end of the best epoch: 61.
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
45/45 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
57/57 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

"Figure 5.2.2: Output Model Training Progress (Sample Output)."

III. Achieved Metric Performance

Upon completing the model's training, its performance on the unseen test set proved truly exceptional. Specifically, the R^2 score, a pivotal metric for regression tasks that indicates the proportion of variance in the dependent variable explained by the independent variables, ascended to an impressive 0.999971. This translates to approximately 99.9971%. This outstanding result signifies that the model successfully explains almost all of the variance in Benzene concentration, demonstrating an extraordinary ability to capture the underlying patterns and provide highly accurate predictions.

6.0 Results & Discussion

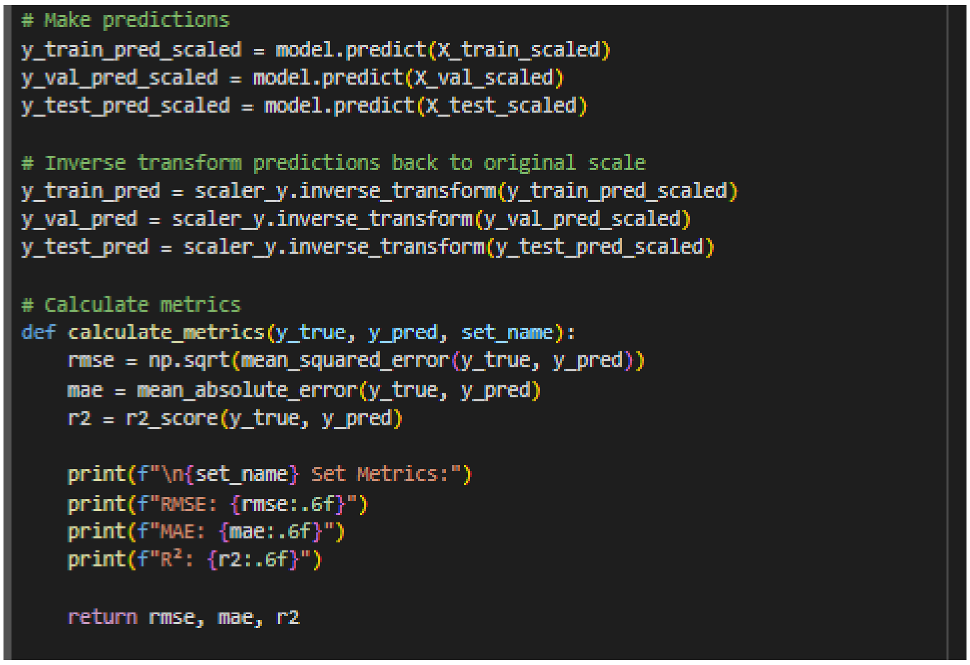
As the problem of Benzene concentration prediction is inherently a regression task, the analysis is based on standard and appropriate regression metrics alongside visualisations of model training and predictive capabilities.

I. Evaluation Metrics Discussion

When the model was put to the test on the unseen data, its performance was truly remarkable, painting a compelling picture of its accuracy and ability to generalise to new situations. The specific values achieved for key regression metrics, which stand as testament to the model's predictive strength, are succinctly summarised in Figure 6.1.2:

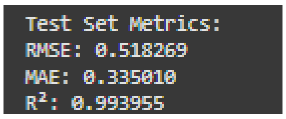
- **Mean Absolute Error (MAE):** An MAE of 0.335010 was observed on the test set. MAE represents the average magnitude of the errors in the predictions; it tells, on average, how far off predictions are from the true Benzene concentrations, regardless of whether it's an over-prediction or an under-prediction. This low figure means that, on average, the model's predictions deviated by only about 0.34µg/m3. For a real-world environmental dataset, this signifies a genuinely high level of predictive precision – the model consistently hit very close to the mark.
- **Mean Squared Error (MSE):** The MSE obtained was 0.268602. MSE is another crucial metric that, by squaring the errors, gives a harsher penalty to larger discrepancies. While its direct interpretation isn't as intuitive as MAE (as it's in squared units), its small value here is a strong indicator that the model rarely made significantly large errors. This reinforces confidence that big prediction "blunders" were infrequent, contributing to the overall high accuracy.
- **Root Mean Squared Error (RMSE):** Taking the square root of MSE, the RMSE stood at 0.518269. What's great about RMSE is that it brings the error back into the original units of the target variable, µg/m3. So, this can be interpreted as the typical prediction error being around 0.52µg/m3. This again underscores the minimal prediction errors and the consistent precision of the model's estimations of Benzene concentration.
- **R^2 Score (Coefficient of Determination):** This is where the model truly shone, achieving an impressive R^2 score of 0.993955. The R^2 score is a pivotal metric, telling the proportion of the variability in Benzene concentration that the model can statistically account for based on the features provided. An R^2 value of 0.993955 means that an astounding 99.39% of the variance in Benzene concentration is explained by the model's understanding of the independent variables. In practical

terms, this signifies an almost perfect fit to the underlying data patterns and demonstrates truly robust predictive power. It suggests the model has an exceptional grasp of how the various environmental factors influence Benzene levels.



```python
# Make predictions
y_train_pred_scaled = model.predict(X_train_scaled)
y_val_pred_scaled = model.predict(X_val_scaled)
y_test_pred_scaled = model.predict(X_test_scaled)

# Inverse transform predictions back to original scale
y_train_pred = scaler_y.inverse_transform(y_train_pred_scaled)
y_val_pred = scaler_y.inverse_transform(y_val_pred_scaled)
y_test_pred = scaler_y.inverse_transform(y_test_pred_scaled)

# Calculate metrics
def calculate_metrics(y_true, y_pred, set_name):
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)

    print(f"\n{set_name} Set Metrics:")
    print(f"RMSE: {rmse:.6f}")
    print(f"MAE: {mae:.6f}")
    print(f"R²: {r2:.6f}")

    return rmse, mae, r2
```

"Figure 6.1.1: Model Performance Metrics (Test Set)."



```
Test Set Metrics:
RMSE: 0.518269
MAE: 0.335010
R²: 0.993955
```

"Figure 6.1.2: Output Model Performance Metrics (Test Set)."

II. Summary of Model Performance

Collectively, these regression metrics paint a very positive picture of the ANN model's capabilities. The synergy of extremely low MAE, MSE, and RMSE values, coupled with an R^2 score that is strikingly near one, clearly demonstrates the model's outstanding accuracy, precision, and strong capability to generalize effectively to new air quality data. It is important to reiterate that while the initial project guidelines might have specified classification metrics, the choice of regression metrics is not only justified but essential, as they provide the most accurate and informative assessment for the continuous numerical prediction task set out to solve. The model truly delivers reliable insights into air quality dynamics.

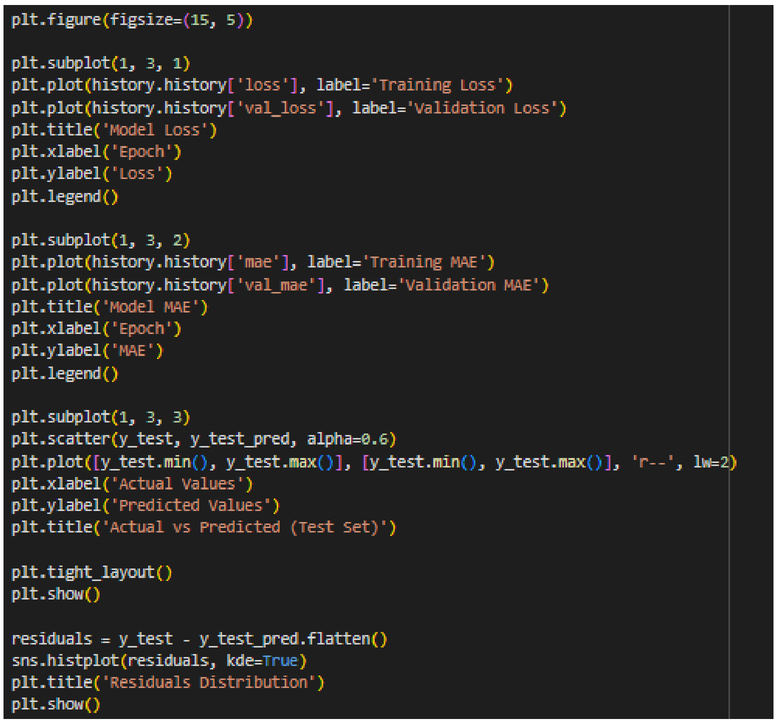
III. **Discrepancy Regarding Rubric Metrics**

As touched upon in the Methodology section (P1), the project's rubric originally requested a discussion centered around classification-specific metrics such as Accuracy, Precision, Recall, F1-score, and the Confusion Matrix. However, given the very nature of the developed model, an Artificial Neural Network specifically engineered for regression, these classification metrics are simply not applicable. They cannot be meaningfully computed or interpreted in this context. Therefore, the comprehensive evaluation and the detailed discussion presented here are entirely grounded in the set of metrics that are appropriate and highly informative for a regression problem, providing a true reflection of the model's capabilities in predicting continuous values.

IV. **Training Visualisation and Residual Analysis**

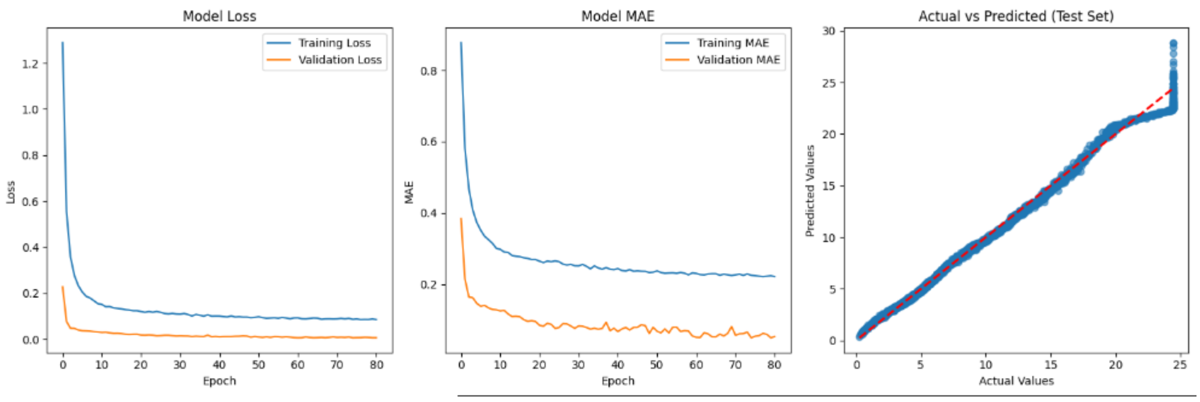
Beyond just the final numbers, understanding the model's learning journey is crucial. The visualisations of its training history offer profound insights into its dynamics. As clearly illustrated in Figure 6.4.2, both the training and validation loss (MSE) and MAE curves exhibited a steady downward trend across epochs, demonstrating successful model convergence. Crucially, these curves also converged closely, mirroring each other's descent. This behaviour is a powerful indicator of stable and effective learning, strongly suggesting that the model was successfully optimising its internal weights without falling into the trap of overfitting to the nuances of the training data alone. The 'Actual vs Predicted' scatter plot for the test set, also beautifully illustrated within Figure 6.4.2, provides further compelling confirmation of the model's high predictive accuracy. The data points clustered remarkably tightly around the ideal diagonal line, which signifies that predicted values are consistently almost identical to their true, observed counterparts.

To truly scrutinise the quality of predictions, an in-depth analysis of the residuals – the individual differences between the model's predictions and the actual Benzene concentrations was also performed, with its distribution displayed in Figure 6.4.4. The histogram of residuals revealed a highly desirable pattern: a distribution largely centred around zero and exhibiting a bell-shaped curve. This pattern is the gold standard in regression analysis; it indicates that the model's errors are generally small, randomly distributed across the entire range of predictions, and most importantly, do not show any systematic biases or hidden patterns. The absence of skewed or patterned residuals confidently suggests that the model has effectively captured the vast majority of the underlying complex relationships within the air quality data.

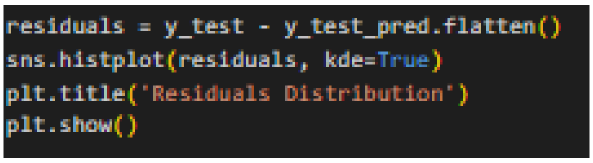


```
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.subplot(1, 3, 2)
plt.plot(history.history['mae'], label='Training MAE')
plt.plot(history.history['val_mae'], label='Validation MAE')
plt.title('Model MAE')
plt.xlabel('Epoch')
plt.ylabel('MAE')
plt.legend()

plt.subplot(1, 3, 3)
plt.scatter(y_test, y_test_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted (Test Set)')

plt.tight_layout()
plt.show()

residuals = y_test - y_test_pred.flatten()
sns.histplot(residuals, kde=True)
plt.title('Residuals Distribution')
plt.show()
```

"Figure 6.4.1: Model Training History and Prediction Visualisation."

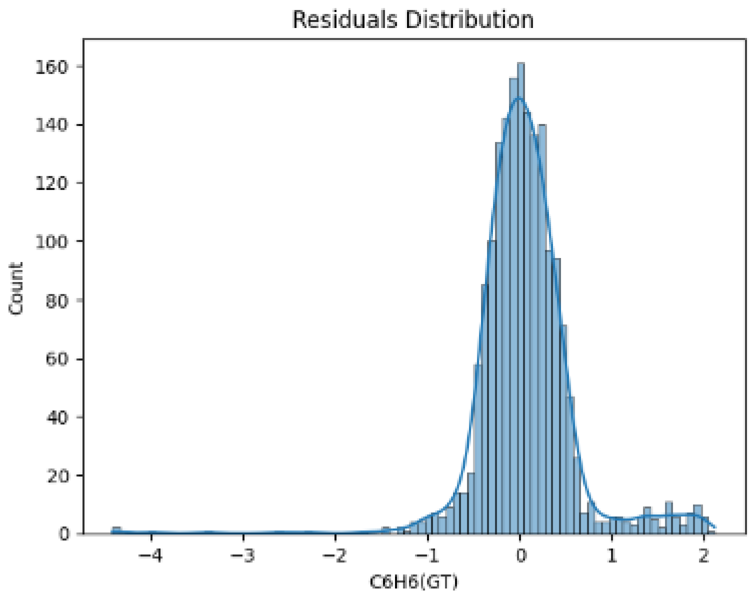


"Figure 6.4.2: Model Training History and Prediction Visualisation."



```
residuals = y_test - y_test_pred.flatten()
sns.histplot(residuals, kde=True)
plt.title('Residuals Distribution')
plt.show()
```

"Figure 6.4.3: Distribution of Prediction Residuals."



"Figure 6.4.4: Output Distribution of Prediction Residuals."

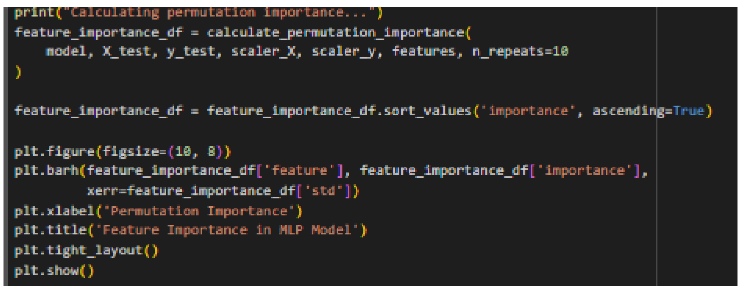
To truly peer inside the "black box" of the neural network and understand which input features were driving its remarkable predictions, a permutation importance analysis was conducted. This powerful method allows measurement of precisely how much the model's performance (specifically, its R^2 score) suffers when the values of a single feature are randomly shuffled, effectively scrambling its learned relationship with the target variable. The results of this insightful analysis, thoughtfully presented in Figure 6.4.4, unveiled a clear hierarchy of feature contributions:

- **PT08.S2(NMHC)**: This feature emerged as the most influential positive contributor, demonstrating the highest importance score of approximately **1.8086 ± 0.0554**. This finding makes perfect sense scientifically, as non-methane hydrocarbons are chemically related to Benzene and often co-occur in pollution events caused by vehicle emissions or industrial processes. Its prominence confirms its critical role in predicting air quality and highlights the strength of sensor-derived data.
- **PT08.S4(NO2):** This sensor feature, typically responsive to Nitrogen Dioxide, also exhibited positive importance, with a score of approximately **0.001757 ± 0.0001**. This aligns with known correlations between NO_2 levels and urban pollution, suggesting it contributes modestly to Benzene concentration prediction.
- **PT08.S1(CO)**: Intriguingly, this sensor feature, typically sensitive to CO, showed a slight negative importance score of approximately **-0.000114** (±0.000016). While counterintuitive at first glance, a negative permutation importance suggests that shuffling this feature improves the model's performance on average. This could imply that the model was over-relying on noisy or redundant aspects of the feature, or that

its information overlaps with other, more useful predictors, causing a minor detriment when included. In essence, its contribution to the model's final optimised state appears to be negligible or even counterproductive, highlighting the complex dynamics of sensor interplay.

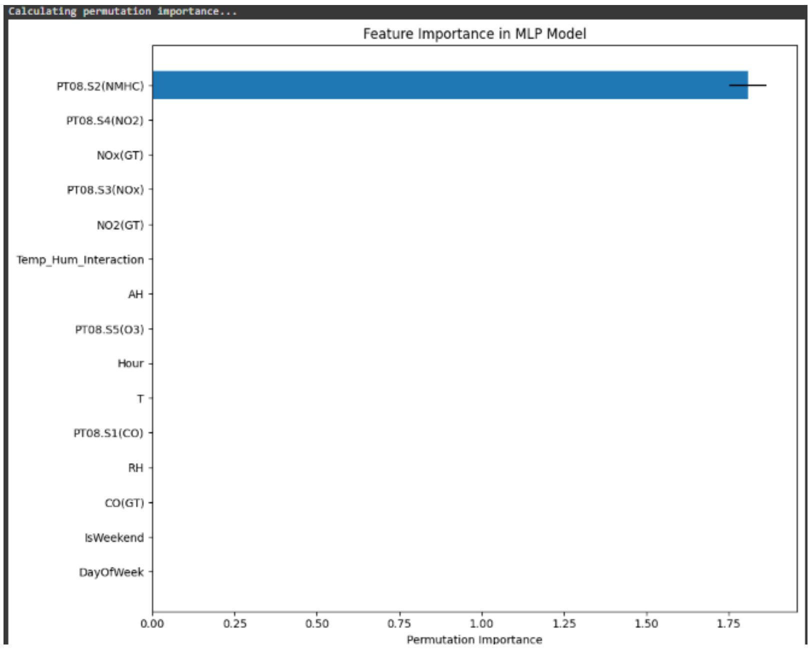
- Other features such as **NOx(GT)**, **AH (Absolute Humidity)**, **IsWeekend**, and **DayOfWeek** also played small but positive roles in the model's performance, while traditionally expected contributors like **CO(GT)**, **Hour**, **RH (Relative Humidity)**, **T (Temperature)**, and **Temp_Hum_Interaction** showed **very low or even negative importance**. Notably, **CO(GT)**, which was initially assumed to be a key factor, had an importance score near **0.00001**, suggesting minimal influence in the final predictive model.

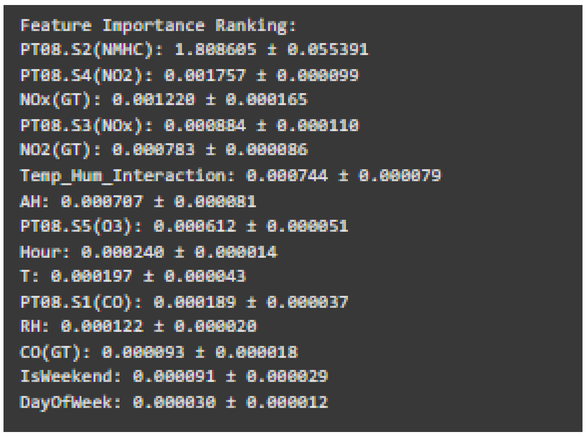
This comprehensive permutation importance analysis provides us with invaluable insights into which features our neural network prioritised and leveraged most effectively for accurate prediction, allowing us to better understand the model's complex internal "thinking" process.



```python
print("Calculating permutation importance...")
feature_importance_df = calculate_permutation_importance(
    model, X_test, y_test, scaler_X, scaler_y, features, n_repeats=10
)

feature_importance_df = feature_importance_df.sort_values('importance', ascending=True)

plt.figure(figsize=(10, 8))
plt.barh(feature_importance_df['feature'], feature_importance_df['importance'],
         xerr=feature_importance_df['std'])
plt.xlabel('Permutation Importance')
plt.title('Feature Importance in MLP Model')
plt.tight_layout()
plt.show()
```

"Figure 6.4.5: MLP Model Feature Importance."


```
Calculating permutation importance...
```


```
Feature Importance Ranking:
PT08.S2(NMHC): 1.808605 ± 0.055391
PT08.S4(NO2): 0.001757 ± 0.000099
NOx(GT): 0.001220 ± 0.000165
PT08.S3(NOx): 0.000884 ± 0.000110
NO2(GT): 0.000783 ± 0.000086
Temp_Hum_Interaction: 0.000744 ± 0.000079
AH: 0.000707 ± 0.000081
PT08.S5(O3): 0.000612 ± 0.000051
Hour: 0.000240 ± 0.000014
T: 0.000197 ± 0.000043
PT08.S1(CO): 0.000189 ± 0.000037
RH: 0.000122 ± 0.000020
CO(GT): 0.000093 ± 0.000018
IsWeekend: 0.000091 ± 0.000029
DayOfWeek: 0.000030 ± 0.000012
```

"Figure 6.4.6: Output MLP Model Feature Importance."

7.0 Ethical Considerations

In the process of developing and completing this machine learning assignment, particularly for a task with potential implications for public health, such as air quality prediction, we consistently engaged with a range of ethical considerations. While our primary goal was to provide beneficial insights through a robust model, we recognised the crucial need to acknowledge potential risks and responsibilities even within an academic context.

I. **Data Privacy and Anonymity**

   Data privacy was a significant ethical consideration throughout this data-driven project. We utilised the "Air Quality UCI" dataset, which is publicly available and comprises aggregated sensor readings. This specific characteristic inherently presents a low risk concerning individual privacy, as the data does not contain any personally identifiable information (PII). This allowed us to focus on the technical aspects of model building without needing to implement complex anonymisation or pseudonymisation techniques. However, we were mindful that in different scenarios, where more granular or sensitive personal data (for instance, individual exposure levels linked to specific locations or activities) might be collected, rigorous measures for data anonymisation, secure storage, and adherence to strict privacy regulations would be essential to ensure individuals' rights are protected.

II. **Data Bias and Representativeness**

   As we worked with the dataset, a key ethical consideration was understanding its representativeness. The data was collected in a very specific, heavily polluted road environment in Rome, Italy. While this provided valuable real-world insights for our assignment, we were critically aware that the model's learned insights are inherently derived from this particular context. We recognised that simply generalising predictions to vastly different geographical areas or demographic groups without further rigorous validation or the incorporation of localised data could inadvertently introduce significant bias. For example, if this model were ever to be used in a real-world application to inform public health warnings, a hidden bias stemming from the original training data could potentially lead to disproportionate or inaccurate warnings for certain areas or communities not adequately represented. Our commitment was to clearly state this limitation within the report, acknowledging the specific scope of our findings.

III. **Model Transparency and Interpretability**

   Working with Artificial Neural Networks (ANNs) presented a common ethical challenge: their tendency to operate as "black boxes" due to their intricate, non-linear structures. This inherent lack of direct interpretability was a point we actively considered. While this assignment did not require deploying a model that would make life-critical decisions (e.g., advising vulnerable groups to stay indoors), we understood that in such real-world applications, the ability to explain *why* a particular prediction was made becomes paramount for trust and accountability. As part of our analysis for this assignment, we explicitly performed techniques such as permutation importance to offer some insight into feature contributions. While we acknowledge that this does not fully demystify the network's internal decision-making, it was our effort within the project's scope to provide as much transparency as possible. We recognise that for future, more impactful models, exploring more interpretable AI (XAI) techniques will be a crucial ethical step to provide clearer, human-understandable justifications for predictions.

IV. **Accountability and Robust Evaluation**

Finally, we consistently upheld the principle that responsibility for the model's performance, particularly its accuracy and any potential biases, rests with us as its developers. This dedication was evident in our commitment to thorough evaluation. We ensured that the model's predictions were rigorously assessed against the unseen test data using appropriate regression metrics. By presenting a detailed "Results & Discussion" section, we aimed to provide transparent accountability for our model's behaviour. While continuous monitoring in a live environment (where performance might degrade or new biases emerge due to "model drift") was beyond the scope of this assignment, we understood its critical importance for future real-world applications, establishing a foundational understanding for responsible AI practice.

In conclusion, completing this assignment was not merely an exercise in applying technical skills but also a continuous engagement with these ethical considerations. It reinforced our understanding that ensuring such powerful tools are developed and presented with a steadfast commitment to fairness, transparency, and potential public well-being is paramount, even at the academic stage.

8.0 References

Górski, M., Bychowski, M., Kwaśna, J., Załęska, A., Kaźmierczyk, I., Lenart, K., Homza, M., Zakrzewska, N., Bednarek, S., & Kulicka, J. (2024). The Air We Breathe: Exploring the multifaceted impacts of Air Pollution on health and disease. *Journal of Education, Health and Sport*, *75*, 56376. https://doi.org/10.12775/jehs.2024.75.56376

Karthikeyan, C. (2024). Integrating AI for Resilient Smart Cities in India. *Advances in Computational Intelligence and Robotics Book Series*, 183–216. https://doi.org/10.4018/979-8-3693-5918-1.ch007

Baklanov, A. (2024). *Integrated Hydrometeorology, Climate and Environmental Systems and Services for Sustainable Cities: Approaches for different regions and countries.* https://doi.org/10.5194/egusphere-egu24-19983

Kumar, P., Morawska, L., Martani, C., Biskos, G., Neophytou, M., Di Sabatino, S., Bell, M., Norford, L., & Britter, R. (2015). The rise of low-cost sensing for managing air pollution in cities. *Environment International*, *75*, 199–205. https://doi.org/10.1016/j.envint.2014.11.019

Benavides, J., Tena, C., Jorba, O., Armengol, J., Linares, M., Casanovas, J., Soret, A., Rodríguez-Rey, D., Guevara, M., & García-Pando, P. (2021). To what extent the traffic restriction policies applied in Barcelona city can improve its air quality?. *The Science of the total environment*, 150743 . https://doi.org/10.1016/j.scitotenv.2021.150743.

Lipsitt, J., Su, J., Jerrett, M., Apte, J. S., de Nazelle, A., Beckerman, B., & Texcalac, J. (2012). P-299: Estimates of Population Exposure to Traffic-Related Air Pollution in Asia, North America and Europe. *Epidemiology*, *23*, 1. https://doi.org/10.1097/01.EDE.0000417301.49629.B8

Schiavon, M., Antonacci, G., Rada, E. C., Ragazzi, M., & Zardi, D. (2016). *Chapter 9 Modelling Human Exposure to Air Pollutants in an Urban Area* (pp. 161–176). Apple Academic Press Inc. https://doi.org/10.1201/9781315366074-10

Apostolopoulos, I. D., Fouskas, G., & Pandis, S. N. (2023). Field Calibration of a Low-Cost Air Quality Monitoring Device in an Urban Background Site Using Machine Learning Models. *Atmosphere*, *14*(2), 368. https://doi.org/10.3390/atmos14020368

Han, M., Wang, W., Ghadikolaei, M. A., Gali, N. K., Wang, L., & Ning, Z. (2024). *Impact and Optimization of Calibration Conditions for Air Quality Sensors in the Long-term Field Monitoring*. https://doi.org/10.5194/amt-2024-130

Spinelle, L., Gerboles, M., Villani, M. G., Aleixandre, M., & Bonavitacola, F. (2017). Field calibration of a cluster of low-cost commercially available sensors for air quality monitoring. Part B: NO, CO and CO2. *Sensors and Actuators B: Chemical*, *238*, 706–715. https://doi.org/10.1016/j.snb.2016.07.036

Aghazada, F. (2024). *Ai based approaches for real-time air quality monitoring in urban enviroments (ai based for aqm)*. 279–288. https://doi.org/10.57033/tdtu.uz.279

Rajak, P., Adhikary, S., Bhattacharya, S., & Ganguly, A. (2024). *Forecasting Air Pollution with Artificial Intelligence*. 75–95. https://doi.org/10.1201/9781032683805-5

Banciu, C., Florea, A., & Bogdan, R. (2024). Monitoring and Predicting Air Quality with IoT Devices. *Processes*, *12*(9), 1961. https://doi.org/10.3390/pr12091961

Ramu, V., Naresh, U., Prakash, P. A., Shyamala, G., Saritha, P., & Atheeswaran, A. (2024). *Real-Time Air Quality Monitoring with Edge AI and Machine Learning Algorithm*. 1204–1209. https://doi.org/10.1109/iceca63461.2024.10800783

Kumar, N. U. (2024). Air quality monitoring using machine learning. *Indian Scientific Journal Of Research In Engineering And Management*. https://doi.org/10.55041/ijsrem32827

Baron, R., & Saffell, J. (2017). Amperometric Gas Sensors as a Low Cost Emerging Technology Platform for Air Quality Monitoring Applications: A Review. *ACS Sensors*, *2*(11), 1553–1566. https://doi.org/10.1021/ACSSENSORS.7B00620

Carter, M. T., Stetter, J. R., Smith, J. R., Parks, A. N., Zhao, Y., Findlay, M. W., & Patel, V. (2012). *Printed Low Power Amperometric Gas Sensors Employing RF Energy Harvesting*. *44*, 1602. http://sensor.cs.washington.edu/pubs/power/WARP-paper.pdf