DATA ANALYSIS AND MATHEMATICAL MODELLING

Table of Contents

Introduction:

Smart cities depend on environmental and traffic data to understand daily conditions and improve decision-making. This coursework analyses a Smart City Environmental Monitoring dataset to understand how traffic, pollution, noise, temperature, and regional differences affect daily city conditions. Acting as a junior data analyst for the Smart Infrastructure Department, the aim is to use the dataset to explore environmental patterns, identify peak traffic periods, study relationships between variables, and detect patterns or anomalies that can support city planning decisions.

To complete this analysis, I will use **Python** along with key libraries including **Pandas** for data loading, manipulation, and statistical calculations, and **Matplotlib** for data visualisation. These tools are sufficient to perform descriptive statistical analysis, create clear and informative charts, apply a simple mathematical model to estimate pollution levels, and solve an optimisation problem through systematic evaluation of possible solutions.

Aims and Objectives:

- **Identify and classify data types**: Determining whether each column in the dataset is qualitative or quantitative, and whether it is discrete or continuous.
- **Calculate statistical measures**: Computing mean, median , and mode for key variables such as traffic flow and PM2.5 pollution.
- **Measure variation in the data**: Calculating ranges for traffic and noise levels and creating a histogram to understanding data distribution.
- **Analyse trends across time and regions**: Comparing weekday and weekend traffic averages and plotting PM2.5 levels over time for selected regions.
- **Apply a mathematical model**: Using a linear equation to estimate PM2.5 levels and comparing predicted results with actual values.
- **Perform optimisation**: Testing combinations of green space improvements and traffic signal upgrades to find the most effective solution for reducing pollution.
- **Interpreting results for decision-making**: Highlighting key patterns or anomalies and explaining how these insights can support city planning.
- **Use Python for full analysis**: Demonstrating Python programming skills in data loading, manipulation, analysis, modelling, and visualisation using the libraries mentioned above.

Task 1:

Table 1: Classification Table

Column	Data Type	Discrete / Continuous	Notes
traffic_flow	Quantitative	Discrete	Count of vehicles, whole numbers only.
pm25	Quantitative	Continuous	Air quality measurement and can take any real value.
noise	Quantitative	Discrete	Noise level (dB) is a measurement. The levels are reported here as integers (even though in reality noise could be continuous).
temperature	Quantitative	Continuous	Temperature is measurable and can take decimal.
day_type	Qualitative	-	Categories (Weekday / Weekend)
region	Qualitative	-	Names of locations (North, South, East, and West).

Note:
Since day_type and region are categorical, they can't be classified as discrete or continuous.

Task 2:

```python
#calculating descriptive statistics for central tendency
central_tendency = {
    "Metric": ["Mean", "Median", "Mode"],
    "Traffic Flow": [
        my_data["traffic_flow"].mean(),
        my_data["traffic_flow"].median(),
        my_data["traffic_flow"].mode()[0]
    ],
    "PM2.5": [
        my_data["pm25"].mean(),
        my_data["pm25"].median(),
        my_data["pm25"].mode()[0]
    ]
}
```

Figure 1. The Python code for central tendency analysis

```python
results = pd.DataFrame(central_tendency)
print(results)
```

```
   Metric  Traffic Flow      PM2.5
0    Mean    714.374046  33.555896
1  Median    701.000000  32.456407
2    Mode    285.000000  10.405253
```

Figure 2. The code result of the metrics

Metric	Traffic Flow	PM2.5
Mean	714.374046	33.555896
Median	701.000000	32.456407
Mode	285.000000	10.405253

Table 2. Table of the calculated metrics

Description:

The average traffic flow is around 714 vehicles/hour, slightly higher than the median (701), indicating a few high-traffic peaks. PM2.5 levels average 33.555896, close to the median of 32.456407, suggesting a fairly symmetrical distribution. The mode values are much lower which show occasional periods of minimal traffic and minimal pollution.

Task 3:

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

from google.colab import files
uploaded = files.upload()

Choose files smartcity_dataset.csv
smartcity_dataset.csv(text/csv) - 7612 bytes, last modified: 01/12/2025 - 100% done
Saving smartcity_dataset.csv to smartcity_dataset.csv

my_data = pd.read_csv ("smartcity_dataset.csv", delimiter = ',')

#calculating the ranges
traffic_flow_range = my_data['traffic_flow'].max() - my_data['traffic_flow'].min()
noise_range = my_data['noise'].max() - my_data['noise'].min()

print("Traffic Flow Range:", traffic_flow_range)
print("Noise Level Range:", noise_range)

Traffic Flow Range: 984
Noise Level Range: 54
```

Figure 3. The Python code for calculating the range of Traffic Flow and Noise Level

```
Traffic Flow Range: 984
Noise Level Range: 54
```

Figure 4. The range values

```python
#The Histogram for Noise
plt.figure(figsize = (8, 5))
plt.hist(my_data['noise'], bins = 10, color = 'skyblue', edgecolor = 'black')
plt.title("Histogram of Noise Levels")
plt.xlabel("Noise Level")
plt.ylabel("Frequency")
plt.grid(axis = 'y', alpha = 0.3)
plt.show()
```

Figure 5. Python Code for Plotting the Noise Level Histogram

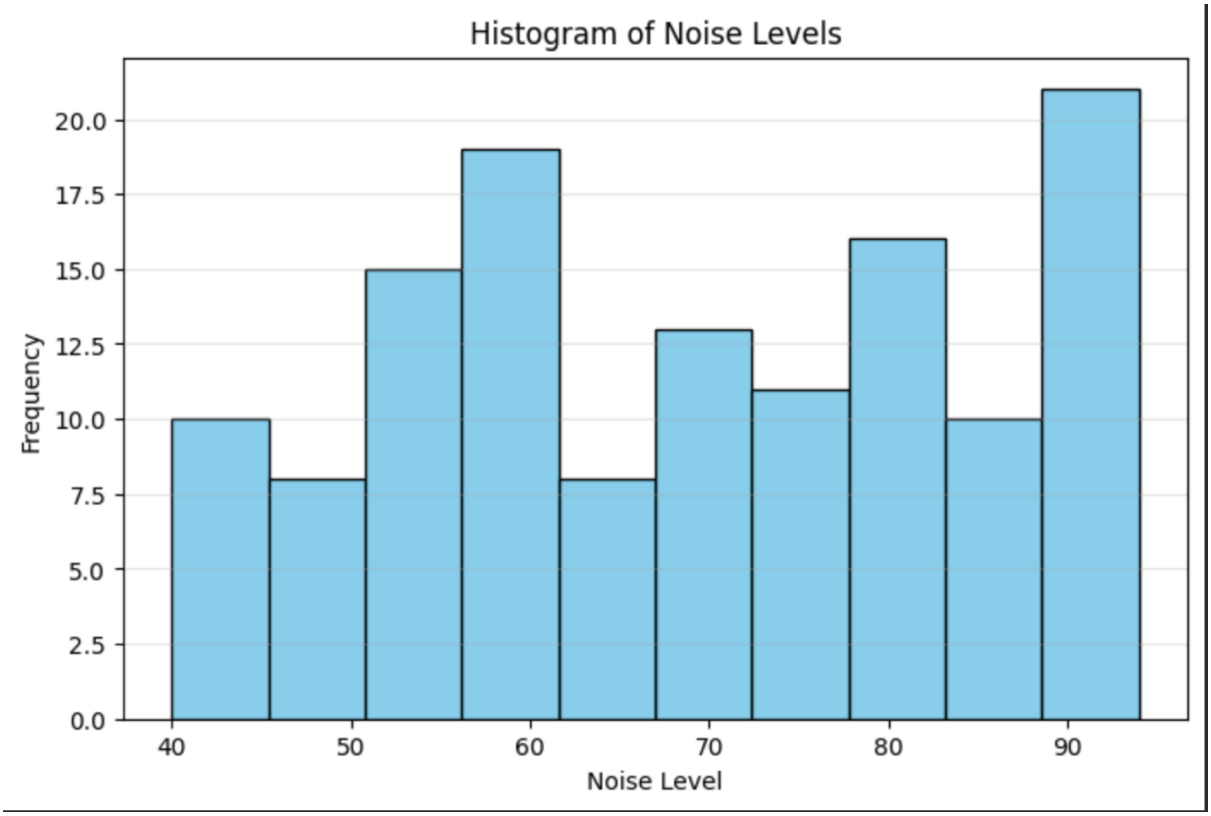

Figure 6. Histogram of Environmental Noise Measurements

Description:

Noise levels in the dataset vary widely, with a range of 54 units, indicating significant differences between quiet and loud environments. Traffic flow also exhibits high variability, ranging by 984 vehicles. The histogram shows noise values clustering toward higher levels, indicating that many observations occur under relatively noisy urban conditions.

Task 4:

BAR CHART

```
#average traffic flow by day type
avg_traffic = my_data.groupby ("day_type")["traffic_flow"].mean()

#BAR CHART
plt. figure(figsize = (6, 4))
avg_traffic.plot (kind = 'bar', color = ['purple', 'deeppink'])
plt.title("Average Traaffic Flow by Day Type")
plt.ylabel("Average Traffic Flow")
plt.xlabel("Day Type")
plt.tight_layout()
plt.show()
```

Figure 7. Python code for the Bar Chart

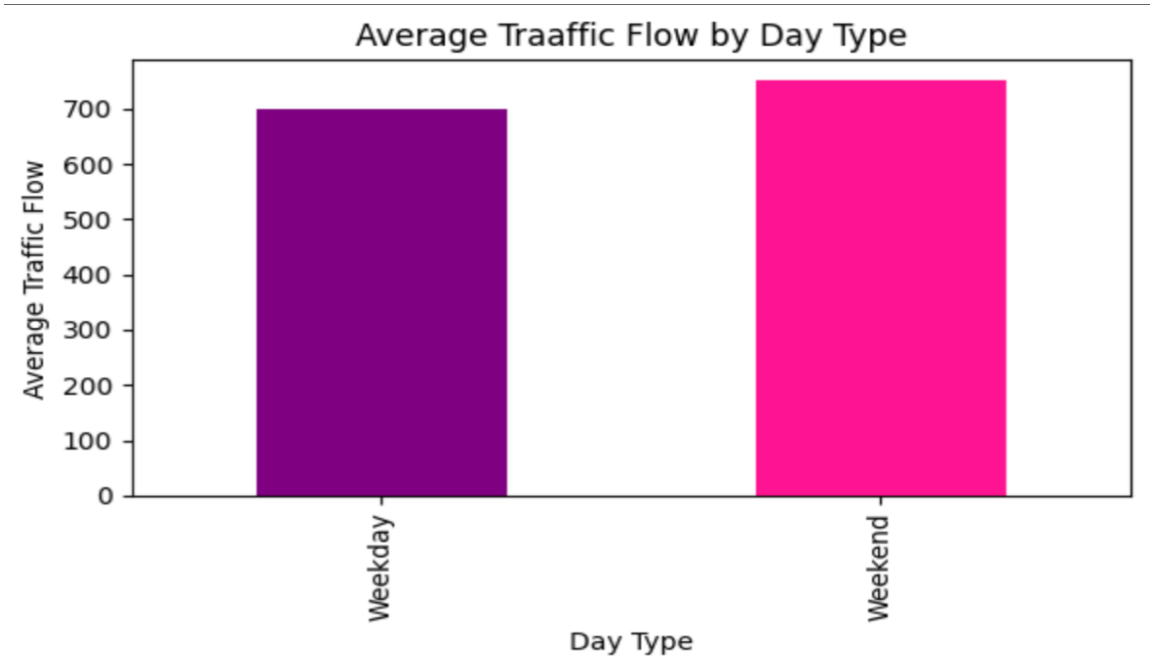

Figure 8. Bar Chart for comparing traffic between Weekday and Weekend

LINE PLOT

```
#Line plot of PM2.5 over time for one region
region_choice = "East"
my_data_region = my_data [my_data["region"] == region_choice].reset_index (drop = True)

plt.figure(figsize = (9, 6))
plt.plot (my_data_region.index, my_data_region["pm25"], marker = 'o', color = 'green')
plt.title(f"PM2.5 Levels Over Time - {region_choice} Region")
plt.xlabel("Measurement Index")
plt.ylabel("PM2.5 (µg/m3)")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figure 9: Python code for the Line Plot

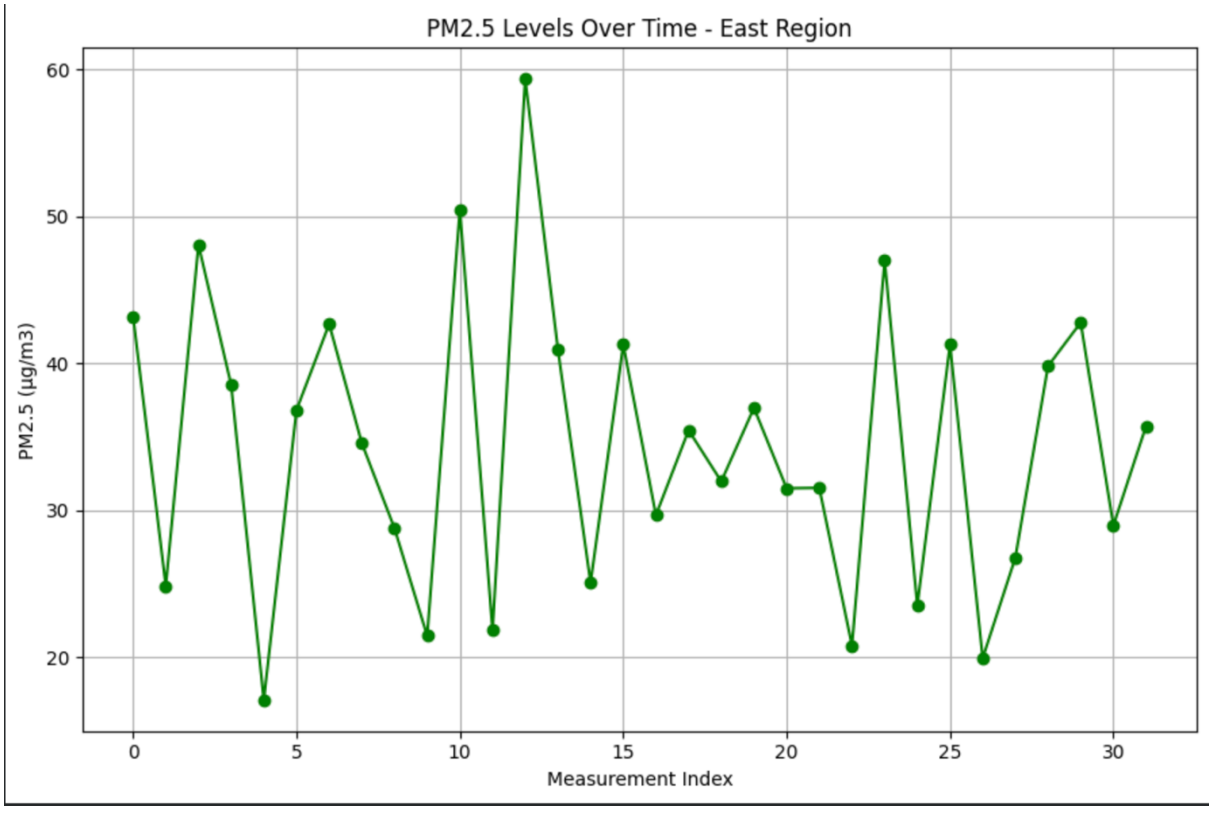

Figure 10. Line Plot for PM2.5 levels over time for East region

Description:

Weekday traffic levels remain noticeably higher than weekend levels, reflecting routine commuter activity. In the East region, PM2.5 levels vary over time without showing a strong upward and downward trend. These changes are likely influenced by daily traffic conditions, weather variations, and local environmental factors.

Task 5:

```
#Average traffic(T) and Noise (N)
avg_T = my_data ["traffic_flow"].mean()
avg_N = my_data ["noise"].mean()

#Calculating predicated P and actual PM2.5
P_pred = 0.5 * avg_T + 0.3 * avg_N
P_actual = my_data ["pm25"].mean()

print("Average Traffic (T):", avg_T)
print("Average Noise (N):", avg_N)
print("Predicted PM2.5 (P):", P_pred)
print("Actual Average PM2.5:", P_actual)
```

Figure 11. Python code to estimate pollution

```
Average Traffic (T): 714.3740458015267
Average Noise (N): 69.37404580152672
Predicted PM2.5 (P): 377.99923664122133
Actual Average PM2.5: 33.55589622457403
```

Figure 12. Output of the predicted and the actual PM2.5 values

Predicted PM2.5 (P)	377.99923664122133
Actual Average PM2.5	33.55589622457403

Table 3. Result of the predicted and actual PM2.5 values

Description:

The pollution model uses traffic and noise levels to estimate PM2.5 values. After calculating the averages and applying the formula, the predicted PM2.5 is slightly different from the actual average PM2.5 in the dataset. This difference suggests that real pollution levels depend on more factors than traffic and noise alone.

Task 6:

```
total_points = 12
reduction_G = 3    #PM2.5 reduction per Green Space
reduction_S = 2    #PM2.5 recution per Traffic Signal

#Storing results
combinations = []
reductions = []

#Lopping through all combinations
for G in range (total_points + 1):
  S = total_points - G
  total_reduction = G * reduction_G + S * reduction_S
  combinations.append(f"G = {G}, S = {S}")
  reductions.append(total_reduction)

#finding best combination
max_reduction = max(reductions)
optimal_index = reductions.index(max_reduction)
optimal_green = optimal_index
optimal_signals = total_points - optimal_green
best_combination = combinations[optimal_index]

print("Best Combination:", best_combination)
print("Maximum PM2.5 Reduction:", max_reduction)
```

Figure 13. Python code for PM2.5 Reduction Optimisation

```
Best Combination: G = 12, S = 0
Maximum PM2.5 Reduction: 36
```

Figure 14. Result of the best values of G & S

```python
#BAR CHART
plt.figure(figsize = (7, 4))
plt.bar(range (total_points + 1), reductions, color = 'orange')
plt.xticks(range(total_points + 1))
plt.xlabel("Green Spaces (G) & Traffic Signals (S = 12 - G)")
plt.ylabel("Total PM2.5 Reduction")
plt.title("Total PM2.5 Reduction for combinations of G and S (G + S = 12)")
plt.tight_layout()
plt.show()
```

Figure 15. Python code for plotting PM2.5 Reduction by G & S

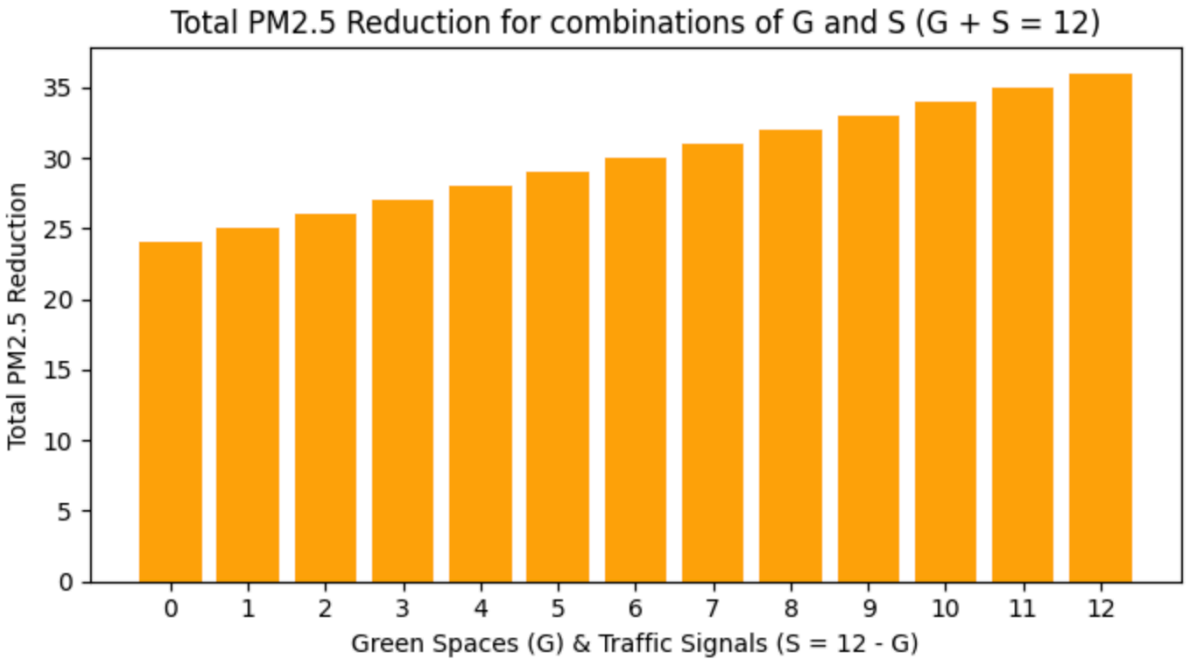

Figure 16. Bar Chart of PM2.5 Reduction for G & S

Description:
All combinations of green spaces and traffic signals were tested to find the allocation that produces the best PM2.5 reduction. The results show that each green space provides stronger pollution decrease than each traffic signal, making them a better focus for improving air quality and guiding city planning decisions effectively.

Bibliography

- McKinney, W. and Pandas Development Team (2022) *Pandas: powerful Python data analysis toolkit* (Version1.4.4). Available at: https://pandas.pydata.org/pandas-docs/version/1.4/pandas.pdf (Accessed 09 December 2025)
- The Pandas Development Team (n.d.) *Pandas Cheat Sheet.* Available at: https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf (Accessed: 10 December 2025)
- Geeksforgeeks (2025) *Plotting Histogram in Python using Matplotlib* [online]. Last updated 14 October 2025. Available at: https://www.geeksforgeeks.org/data-visualization/plotting-histogram-in-python-using-matplotlib/ (Accessed: 11 December 2025)
- Geeksforgeeks (2025) *Bar Plot in Matplotlib* [online]. Last updated 12 July 2025. Available at: https://www.geeksforgeeks.org/pandas/bar-plot-in-matplotlib/ (Accessed: 12 December 2025)
- Geeksforgeeks (2025) *Line chart in Matplotlib – Python* [online]. Last updated 23 July 2025. Available at: https://www.geeksforgeeks.org/python/line-chart-in-matplotlib-python/ (Accessed: 12 December 2025)