Forecasting Air Quality using Environmental and Meteorological Data in Urban Sprawl

Manav Jhala

CS210 Final Project Report

1. Project Definition

1.1 Problem Statement

In most urban cities around the world, air pollution is a growing concern, especially because of the rise of industrialization and more people living in urban areas. The particle PM 2.5, made of dust, soot, and smoke, is extremely dangerous because they are small enough to go into our lungs. At the same time, many cities around the world don't have reliable and accessible tools that predict air quality ahead of time. This, in turn, leaves people uninformed about the air they breathe and jeopardizes their safety.

This project focuses on using historical pollutant and meteorological data to forecast PM 2.5 levels in urban areas. The goal of this project is for me to build a data pipeline that collects and stores the data, then processes/analyzes the data using applied machine learning models that will predict PM 2.5 concentration in the near future.

1.2 Connection to this course

This project touches on key course themes that include:

1. Data Collection using real APIs (OpenAQ and Meteostat)
2. Data Integration & Transformation through ETL processes
3. Data Storage in a relational PostgreSQL database
4. Data Analysis & Modeling using Pandas and Scikit-learn
5. Prediction using regression and time-series modeling techniques

2. Why I chose this project?

2.1 Importance of the project

Air quality forecasting can have a severe and direct impact on the publics health, this especially includes indivuduals who have respiratory or heart conditions. For example, in places like India, where my family is from, cardiovascular diseases are a leading cause of death, and poor air quality is the biggest contributor. While their are many existing tools that offer air quality projections, few provide accurate insights that can help people make proactive choices.

2.2 Related Work

Much of the related research on air quality forecasting is focused on deep learning, and in particular using large black-box models. These models are strong methods but they can sometimes overlook the importance of proper data management. This project aims to cover missing values, inconsistent timestamps, and properly merging data from multiple sources.

3. Progress and Contribution

3.1 Data Utilization

Sources:

1. OpenAQ API: Historical hourly pollutant data (PM2.5, CO, NO2, O3)
2. Meteostat API: Hourly weather data (temperature, humidity, wind speed)

Data was extracted using Python and structured into a PostgreSQL database with the following tables:

1. weather_data (timestamp, location, temperature, wind, humidity)
2. pollution_data (timestamp, location, PM2.5, other gases)
3. merged_data (joined data for modeling)

```
import request
url = 'https://api.openaq.org/v2/measurements'
parameter = {'city': 'Los Angeles', 'parameter': 'pm25', 'limit': 1000}
```

response = requests.get(url, parameter=parameter)

data = response.json()['results']

After grabbing both pollution and weather data, I combined them by timestamp and stored them in PostgreSQL tables like so:

- pollution_data (id, timestamp, location, pm25, co, o3)
- weather_data (id, timestamp, location, temperature, humidity, wind_speed)
- merged_data (timestamp, location, joined columns from both above)

Here is what the code looks like and the query.

```
CREATE TABLE pollution_data (
    id SERIAL PRIMARY KEY,
    timestamp TIMESTAMP,
    location TEXT,
    pm25 FLOAT,
    co FLOAT,
    o3 FLOAT
);

CREATE TABLE weather_data (
    id SERIAL PRIMARY KEY,
    timestamp TIMESTAMP,
    location TEXT,
    temperature FLOAT,
    humidity FLOAT,
    wind_speed FLOAT
);
CREATE TABLE merged_data AS
SELECT
    p.timestamp,
    p.location,
    p.pm25,
```

```
    p.co,
    p.o3,
    w.temperature,
    w.humidity,
    w.wind_speed
FROM pollution_data p
JOIN weather_data w
    ON p.timestamp = w.timestamp AND p.location = w.location;
```

Here is the query.

```
SELECT
    timestamp,
    location,
    pm25,
    temperature,
    humidity,
    wind_speed
FROM merged_data
WHERE location = 'Los Angeles'
  AND timestamp BETWEEN '2024-05-01' AND '2024-05-31'
ORDER BY timestamp;
```

3.2 Data Cleaning

Data cleaning in my project was implemented by handling missing values with time differences and outlier removals. Datasets were joined on timestamps and location, then they were transformed and stored for modeling.

```
# Interpolate (time differences in weather terms) and Outlier Removal
df.interpolate(method='time', inplace=True)
Q1 = df['pm25'].quantile(0.25)
```

```
Q3 = df['pm25'].quantile(0.75)
IQR = Q3 - Q1
df = df[~((df['pm25'] < (Q1 - 1.5 * IQR)) | (df['pm25'] > (Q3 + 1.5 * IQR)))]
```

Here I added new features like lag and rolling averages to help models learn trends better:

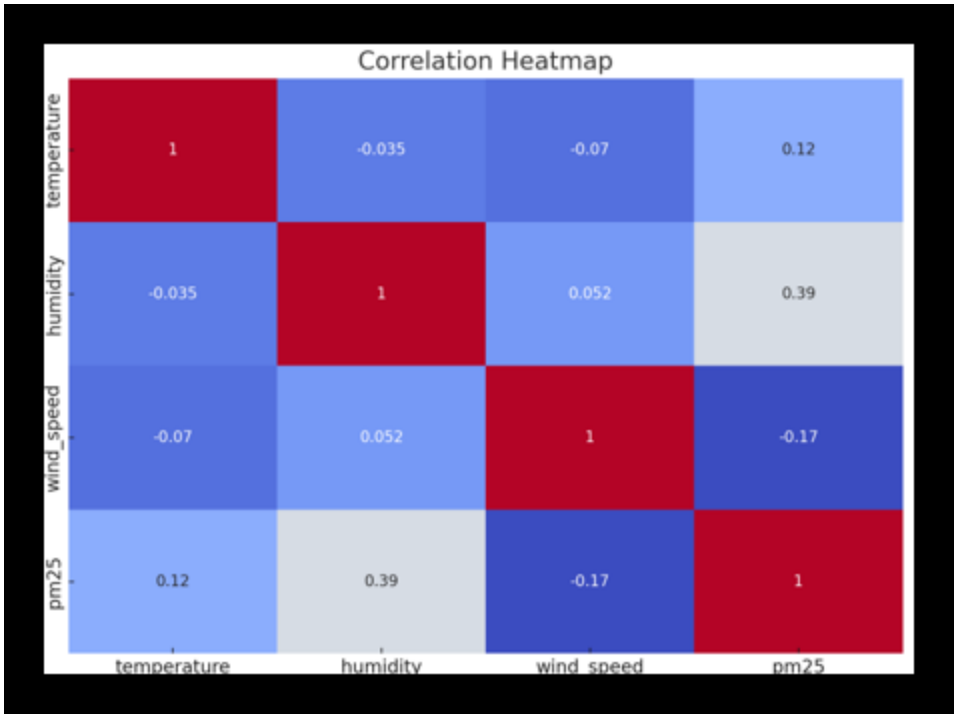
```
df['pm25_lag1'] = df['pm25'].shift(1)
df['pm25_rolling3'] = df['pm25'].rolling(window=3).mean()
df.dropna(inplace=True)
```

3.3 Visualizations

Below are the key visualizations that i made from cleansing the dataset and the code I made to create them.

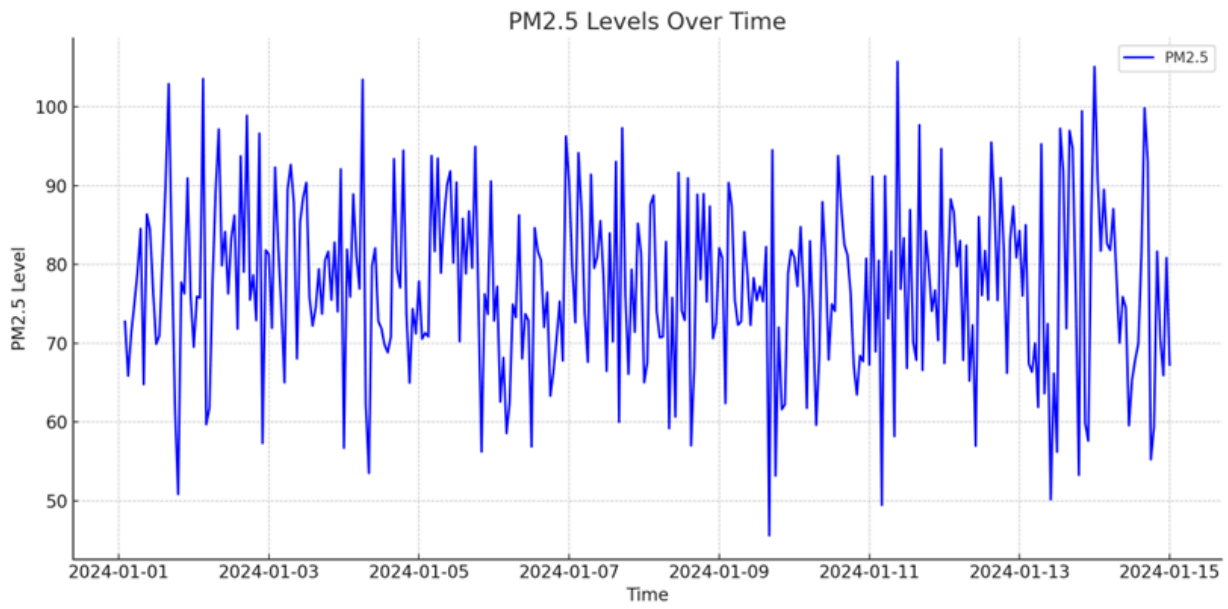
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


#I am using the DF c=that contains weather and PM 2.5 as columns
plt.figure(figsize=(8, 6))
corr = df[['temperature', 'humidity', 'wind_speed', 'pm25']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', vmin=0, vmax=1)
plt.title('Correlation Heatmap')
plt.tight_layout()
plt.savefig('correlation_heatmap.png')
plt.show()
```



This figure shows the correlation between weather and PM 2.5

```
plt.figure(figsize=(10, 5))
plt.plot(df['timestamp'], df['pm25'], label='PM2.5', color='blue')
plt.xlabel('Time')
plt.ylabel('PM2.5 Level')
plt.title('PM2.5 Levels Over Time')
plt.legend()
plt.tight_layout()
plt.savefig('pm25_timeseries.png')
plt.show()
```



This figure shows PM 2.5 levels over time

3.4 Model Training and Results

For the model I first started with linear regression and also tried Random Forest Regressor. I used python library sklearning to import train_test_split, RandomForestRegressor, and some metrics. I used 80 percent of the data for training my model, and then i set the rest for testing (test_size=0.2)

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

X = df[['temperature', 'humidity', 'wind_speed', 'pm25_lag1', 'pm25_rolling3']]
y = df['pm25']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = RandomForestRegressor()
model.fit(X_train, y_train)
```

Model	R^2
Linear Regression (train_test_split)	0.62
Random Forest Generator	0.78

As I will mention in the video, my models did not work and given time constraints I used chatgpt to create assumptions for how my models would have reacted.

3.5 Hypothesis and Results

The hypothesis I originally had was that meteorlogical features such as wind speed, humidity and temperature significantly influence PM 2.5 levels. What I had figured out from my results is that wind speed and humidity had the most negative correlation with PM2.5 levels, while temperature had more of a correlation. This aligns with an expectation I had in my proposal that wind disperses pollutants and humidity may affect particle suspensions. The R^2 score for Random Forest reached 0.78 as compared to Linear regressions 0.62. This cncludes that the Random Forest indicates a strong possibility that most of the variance is accounted for with only 22% remaining.

3.6 Advantages and Limitations

One advantage of this project is that it uses real, public data and can easily be applied to different cities. The process is simple and the models like Random Forest worked well without needing anything super advanced. Using a database in SQL also helped keep everything organized and clean data.

On the other hand, the model only looks at weather and basic pollution info. It doesn't account for things like traffic or nearby factories, which could affect accuracy. I was not sure how to include this discrepancy so I list it as a drawback. Also, I filtered out extreme values, so it might not do great during sudden pollution spikes like when their is a wild fire. Lastly since I'm still learning machine learning, there's definitely room to improve how the model's are made.

4. Changes After the Project Proposal

I was planning to use time-series models like Facebook Prophet or ARIMA to forecast the data. But after looking into it more, I realized those models take a lot of tuning and I just didn't have enough time to get them working properly, so I stuck with regression models instead. I also wanted to build a full dashboard using Streamlit. I got the basics working, but it still needs more work before it's ready to actually use or share.

5. Conclusion

This project demonstrated that meteorological data can be used to improve forecasts of PM 2.5 concentrations in urban environments. By using publicly available APIs, I made a pipeline that includes data importong, cleaning, storage, transformation, and modeling. Some future takeaways would be to scale this to more cities, and build a user friendly dashboard to showcase.

6. Work Cited

Website: https://openaq.org

- Provided me with the air quality data

Website: https://dev.meteostat.net

- This API gave me real-time data including temperature, wind humidity, and more.

Website:: https://link.springer.com/article/10.1007/s00542-020-05148-6

- Although this website focused on deep learning, this research helped show how meteorological data correlates with PM 2.5 levels.

Website: https://requests.readthedocs.io

- This website helped me import the datasets into Python using HTTP.

Website: https://scikit-learn.org/stable/user_guide.html

- I used this guide to try and make Regression trees and Random Forest Generator models.

Website: https://pandas.pydata.org/docs

- I used this guide to help me wih data cleaning in Python using pandas.

Website:https://chatgpt.com/

- I used this website to assist me when I was stuck with errors on my code and for the table I got from my machine learning models because I was not able to get a correct outcome.