AQI Predictor Prototype

Goal: Predict the next 24 hour AQI, based on the past 12-hour data (30 mins-interval)

Last Updated: 10/06/2025

- Bahy Helmi Hartoyo Putra
- bahyhelmi97@gmail.com

Training Pipeline

```
# 1. Import Libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.multioutput import MultiOutputRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
import joblib
import matplotlib.pyplot as plt

# 2. Load Data
df = pd.read_csv("input.csv", parse_dates=['timestamp'])
df = df.sort_values("timestamp")

# For prototype, we'll use: Subset last X days
x_days = 2
df = df[df['timestamp'] >= df['timestamp'].max() -
pd.Timedelta(days=x_days)]
print(df.shape)
df.head(2)
```

```
(97, 37)

                         created_at                          updated_at
id  \
6691  2025-05-24 11:53:53.958527+00  2025-05-29 09:23:37.76187+00
13559
6692  2025-05-24 11:53:53.961339+00  2025-05-29 09:30:38.84316+00
13560

                     timestamp     PM25     PM10       O3      SO2
NO  \
6691 2025-05-22 18:30:00+00:00  17.0167  28.0616  18.6344  6.9171
2.2333
6692 2025-05-22 19:00:00+00:00  26.7487  40.9343  21.5069  6.9354
1.0469

          NO2  ...  AQI
AQI_detail  \
```

```
6691  35.3888  ...   78  {"CO": 21, "O3": 11, "CH4": 4, "NO2": 29,
"SO2...
6692  25.4352  ...   77  {"CO": 21, "O3": 11, "CH4": 4, "NO2": 28,
"SO2...

      AQI_parameter  PM25_AQI  PM10_AQI  O3_AQI  SO2_AQI  NO2_AQI
CO_AQI  \
6691           PM25        78        40      11        9       29
21
6692           PM25        77        40      11        9       28
21

      CH4_AQI
6691        4
6692        4

[2 rows x 37 columns]
```

```python
# 3. Feature Selection
features = ['PM25', 'PM10', 'O3', 'SO2', 'NO', 'NO2', 'NOX', 'CO',
'CH4',
            'NMHC', 'THC', 'wind_speed', 'wind_gust_speed',
            'wind_direction', 'air_humidity', 'air_temperature',
            'container_humidity', 'container_temperature',
'solar_radiation']
target = 'AQI'

# 4. Framing the problem as Supervised Learning
input_len = 24   # past 12 hours (30min × 24 = 12hr)
output_len = 48  # next 24 hours (30min × 48 = 24hr)

X, y = [], []
for i in range(input_len, len(df) - output_len):
    X.append(df[features].iloc[i - input_len:i].values.flatten())
    y.append(df[target].iloc[i:i + output_len].values)

X = np.array(X)
y = np.array(y)

X.shape, y.shape
```

```
((25, 456), (25, 48))
```

```python
# 5. Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, shuffle=False)

# 6. Normalize
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# 7. Model Training
# Caution: Might takes time! Please be patient, monitor your CPU &
Memory usage.
base_model = GradientBoostingRegressor()
model = MultiOutputRegressor(base_model)
model.fit(X_train_scaled, y_train)
```

```
MultiOutputRegressor(estimator=GradientBoostingRegressor())
```

```
# 8. Save Model
joblib.dump(model, 'aqi_forecast_model.pkl')
joblib.dump(scaler, 'aqi_scaler.pkl')
```

```
['aqi_scaler.pkl']
```

Inference Pipeline

```
X_test_scaled.shape
```

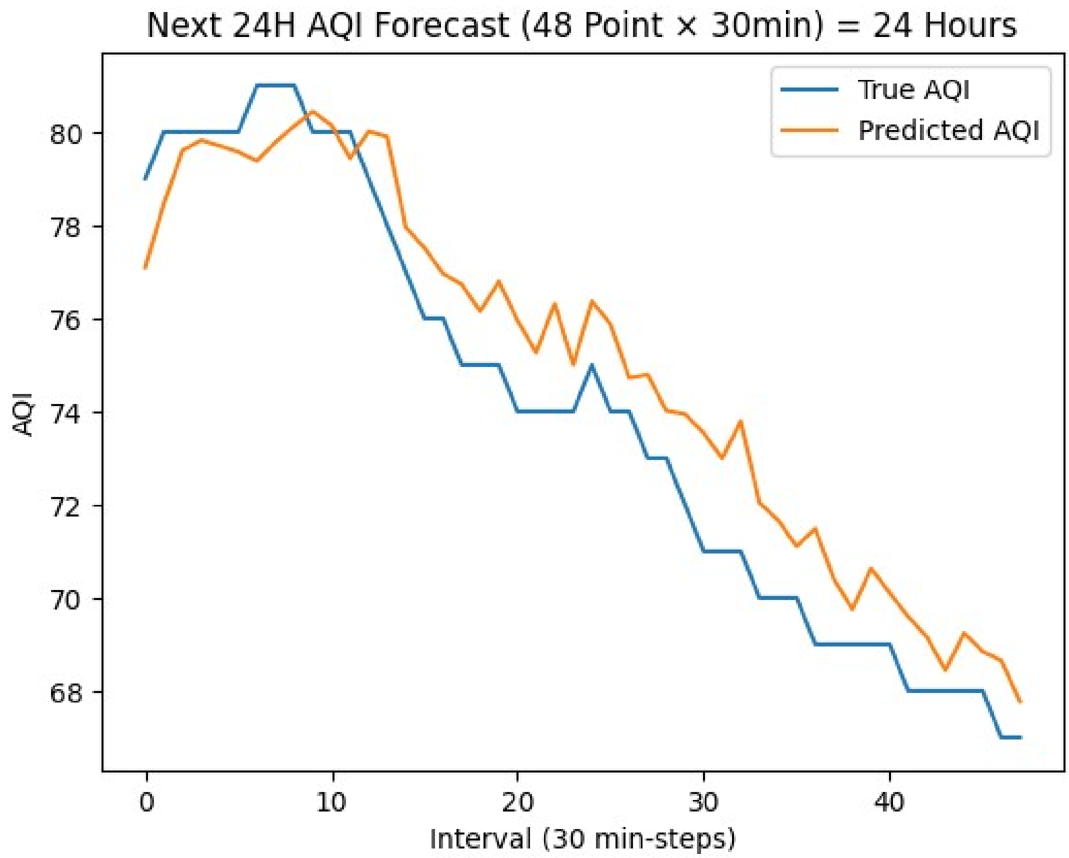
```
(5, 456)
```

```
# 9. Inference
y_pred = model.predict(X_test_scaled)
print("Mean Absolute Error (MAE):", mean_absolute_error(y_test,
y_pred))
```

```
Mean Absolute Error (MAE): 2.427852716662926
```

```
# 10. Visualization Example of the first 24 hours rolling window
plt.plot(y_test[0], label='True AQI')
plt.plot(y_pred[0], label='Predicted AQI')
plt.title('Next 24H AQI Forecast (48 Point × 30min) = 24 Hours')
plt.legend()

plt.xlabel("Interval (30 min-steps)")
plt.ylabel("AQI")
plt.show()
```



Summary

- In this prototype, for the sake of efficiency & proof-of-concept, we only use the most recent **2 days** of data during the **training**.
- We could always increase the time window to (possibly) get more (seasonal/monthly) context.
- Accuracy is not bad, as visualize on the above graph!