AIR QUALITY INDEX FORECASTING SYSTEM FOR KARACHI

1. Introduction

The objective of this project is to design and develop a complete end-to-end Air Quality Index (AQI) forecasting system for Karachi that predicts AQI for three days using Machine Learning techniques and MLOps practices.

This project implements a full production-ready machine learning pipeline that:

- Fetches real-time and historical data from Openmeteo api
- Performs data validation and cleaning on raw data
- Engineer features
- Store features in Hopsworks feature store
- Trains multiple ml models
- Register models in Hopsworks model registry with proper models versioning
- Deploys the model predictions using Streamlit
- Automates feature and training pipeline using CI/CD (Github Actions)

The goal was to design a scalable, automated and reproducible AQI forecasting system.

2. Data Collection and API Selection

2.1 INITIAL CHALLENGE

For me, initially, selecting a suitable API was challenging. The major issues were:

• Many APIs did not provide one year of historical data.
• Some APIs had strict rate limiting.
• Some did not provide complete pollutant data (PM2.5, PM10, NO_2, CO, SO_2, O_3, etc.).
• Some required paid subscriptions for historical access.

After testing multiple APIs, I selected the Open-Meteo API because:

• It provides one year of historical hourly data.
• It includes both AQI pollutants and weather parameters.
• It has manageable rate limits.
• It supports hourly time resolution.

```
self.air_quality_url = "https://air-quality-api.open-meteo.com/v1/air-quality"

self.weather_url = "https://historical-forecast-api.open-meteo.com/v1/forecast"
```

3. Data Fetching and Backfilling

3.1 Real-Time and Historical Data Fetching

I implemented a data fetcher module that retrieves:

• AQI pollutant data (PM2.5, PM10, NO_2, CO, SO_2, O_3)
• Weather data (temperature, humidity, wind speed, wind direction,surface pressure)

Both datasets are fetched and merged based on hourly timestamps.

```
📥 Fetching AIR QUALITY from 2025-01-23 to 2026-01-23
✅ Air-quality rows: 8784
🌦Fetching WEATHER from 2025-01-23 to 2026-01-23
🔗 Merging air-quality + weather data...
🎯 Final merged rows: 8784
            timestamp      pm2_5  ...  wind_direction_10m  surface_pressure
0 2025-01-23 00:00:00  44.500000  ...          345.699677       1015.359497
1 2025-01-23 01:00:00  43.700001  ...          343.412567       1016.257507
2 2025-01-23 02:00:00  45.500000  ...          352.476257       1016.856079
3 2025-01-23 03:00:00  51.599998  ...          358.876709       1017.556152
4 2025-01-23 04:00:00  53.000000  ...            4.398633       1018.459839
```

This confirms correct hourly merging and alignment.

3.2 One-Year Historical Backfill

To build a reliable forecasting system, I performed a one-year historical backfill.

Purpose of backfilling:

• To collect sufficient training data
• To ensure if there are any missing time gaps
• To validate timestamp continuity
• To verify API consistency

```
🔄Starting backfill for last 365 days...
🎯 Final merged rows: 8784
✅ Raw data saved: data/raw/aq_weather_raw.csv
📊 Rows: 8784 | Columns: 13

🔍 Sanity Check:
Date range:
 From: 2025-01-23 00:00:00
 To  : 2026-01-23 23:00:00
```

I also generated a raw CSV file from this backfilled data to manually check:

- Missing timestamps
- Unexpected gaps
- Proper pollutant values
- Correct merging of AQI pollutants and weather data

```
timestamp,pm2_5,pm10,carbon_monoxide,nitrogen_dioxide,sulphur_dioxide,ozone,ammonia,temperature_2m,relative_humidity_2m,wind_speed_10m,wind_direction_10m,surface_pressur
2025-01-23 00:00:00,44.5,49.8,386.0,24.2,18.4,54.0,,15.8,53.968857,9.473541,345.69968,1015.3595
2025-01-23 01:00:00,43.7,48.4,547.0,31.1,19.1,45.0,,15.4,58.291676,8.827343,343.41257,1016.2575
2025-01-23 02:00:00,45.5,49.6,802.0,40.3,20.1,33.0,,15.1,58.818638,9.6228485,352.47626,1016.8561
2025-01-23 03:00:00,51.6,55.3,965.0,44.2,20.8,32.0,,15.35,55.73829,9.181765,358.8767,1017.55615
2025-01-23 04:00:00,53.0,56.7,948.0,38.1,21.4,53.0,,16.85,47.932686,7.040739,4.398633,1018.45984
2025-01-23 05:00:00,45.9,52.6,838.0,26.7,21.7,84.0,,19.45,36.675182,6.0721655,11.976112,1019.067
2025-01-23 06:00:00,32.7,44.9,711.0,17.0,20.8,109.0,,22.0,29.320587,5.6521144,9.16228,1019.07404
2025-01-23 07:00:00,21.7,37.3,567.0,11.1,17.7,120.0,,24.2,23.996792,6.8423676,26.564985,1017.8811
2025-01-23 08:00:00,18.5,34.0,407.0,6.8,13.4,124.0,,25.55,19.180855,6.889557,19.855309,1016.586
2025-01-23 09:00:00,15.8,29.7,302.0,4.3,10.1,126.0,,26.1,16.046677,6.9270773,24.567156,1015.4881
2025-01-23 10:00:00,13.2,25.3,273.0,3.3,8.7,127.0,,26.4,12.514948,8.415842,48.468323,1014.6897
2025-01-23 11:00:00,11.7,22.2,299.0,4.1,8.2,125.0,,26.2,13.348256,10.094454,58.861095,1014.48926
2025-01-23 12:00:00,10.0,15.9,399.0,7.5,7.9,117.0,,25.7,14.48816,9.82114,63.904633,1014.68774
2025-01-23 13:00:00,12.4,16.2,647.0,16.4,8.6,102.0,,24.75,15.273955,12.031756,51.072456,1015.38464
2025-01-23 14:00:00,16.6,19.0,967.0,28.4,9.5,80.0,,23.4,16.687405,10.152064,52.92685,1016.0804
2025-01-23 15:00:00,17.5,19.7,1150.0,36.0,10.1,65.0,,22.2,18.69558,10.486448,39.427776,1016.9763
2025-01-23 16:00:00,16.7,19.2,1074.0,35.3,10.1,61.0,,20.95,21.808851,9.835975,34.56259,1017.4724
```

This raw csv was created only for validation and understanding purposes and was not used anywhere.

4. Data Quality Checks

Before cleaning the dataset, I performed data quality checks on the raw csv file.

Checks included:

- Missing values
- Timelines
- Validity
- Consistency
- Temporal Patterns

```
🔍 DATA QUALITY REPORT
============================================================

1 COMPLETENESS:
   Total records: 8784
   Missing values:
     ammonia: 8784 (100.0%)

2 TIMELINESS:
   Expected gap: 1 hour
```

```
2 TIMELINESS:
   Expected gap: 1 hour
   Gaps found: 1
   ⚠Time gaps detected at 1 locations
```

```
3 VALIDITY:
  pm2_5:
    Min: 3.80
    Max: 108.70
    Mean: 29.30
    ✅ Values within expected range
  pm10:
    Min: 3.90
    Max: 385.60
    Mean: 55.68
    ✅ Values within expected range
```

```
  relative_humidity_2m:
    Min: 4.66
    Max: 99.38
    Mean: 62.81
    ✅ Values within expected range

4 CONSISTENCY:
  PM2.5/PM10 ratio: 0.60
  ✅ Ratio is realistic
```

```
5 TEMPORAL PATTERNS:
  Peak pollution hour: 15:00 (32.9 µg/m³)
  Lowest pollution hour: 0:00 (25.7 µg/m³)
  ⚠Unusual peak timing

============================================================
✅ DATA QUALITY CHECK COMPLETE!
```

These checks ensured temporal consistency, data reliability and model-readiness before moving to data cleaning and feature engineering.

5. Data Cleaning

After validation, I implemented a Data Cleaner module that:

• Handled missing values
• Ensured proper datetime formatting
• Removed inconsistencies
• Standardized column naming
• Verified numerical data types
• Applied outlier capping

```
📥 Loading raw data...
🧹 Starting data cleaning...
⚠Dropping empty columns: ['ammonia']
⚠Capping 404 outliers in pm2_5
⚠Capping 332 outliers in pm10
⚠Capping 547 outliers in carbon_monoxide
⚠Capping 505 outliers in nitrogen_dioxide
⚠Capping 486 outliers in sulphur_dioxide
⚠Capping 90 outliers in ozone
✅ Data cleaning complete
```

```
✅ Data cleaning complete

🔍 DATA QUALITY REPORT
============================================================

1️⃣COMPLETENESS:
  Total records: 8784

2️⃣TIMELINESS:
  Expected gap: 1 hour
  Gaps found: 0
```

```
    Max: 119.80
    Mean: 53.97
    ✅ Values within expected range
  carbon_monoxide:
    Min: 74.00
    Max: 1340.50
    Mean: 514.63
    ✅ Values within expected range
  nitrogen_dioxide:
    Min: 1.50
    Max: 60.75
```

```
    Max: 60.75
    Mean: 20.89
    ✅ Values within expected range
  sulphur_dioxide:
    Min: 4.90
    Max: 32.35
    Mean: 15.71
    ✅ Values within expected range
  ozone:
    Min: 0.00
    Max: 175.50
```

```
    ✅ Values within expected range

4️⃣CONSISTENCY:
  PM2.5/PM10 ratio: 0.53
  ✅ Ratio is realistic

5️⃣TEMPORAL PATTERNS:
  Peak pollution hour: 4:00 (31.74 µg/m³)
  Lowest pollution hour: 0:00 (25.08 µg/m³)

✅ DATA QUALITY REPORT COMPLETE
```

After cleaning, I generated another CSV file to confirm:

• Data cleaning worked correctly

• No unwanted null values remained

• Final dataset structure was correct

```
data > processed > aq_weather_clean.csv
timestamp,pm2_5,pm10,carbon_monoxide,nitrogen_dioxide,sulphur_dioxide,ozone,temperature_2m,relative_humidity_2m,wind_speed_10m,wind_direction_10m,surface_pressure,hour
2025-01-23 00:00:00,44.5,49.8,386.0,24.2,18.4,54.0,15.8,53.968857,9.473541,345.69968,1015.3595,0
2025-01-23 01:00:00,43.7,48.4,547.0,31.1,19.1,45.0,15.4,58.291676,8.827343,343.41257,1016.2575,1
2025-01-23 02:00:00,45.5,49.6,802.0,40.3,20.1,33.0,15.1,58.818638,9.6228485,352.47626,1016.8561,2
2025-01-23 03:00:00,51.6,55.3,965.0,44.2,20.8,32.0,15.35,55.73829,9.181765,358.8767,1017.55615,3
2025-01-23 04:00:00,53.0,56.7,948.0,38.1,21.4,53.0,16.85,47.932686,7.040739,4.398633,1018.45984,4
2025-01-23 05:00:00,45.9,52.6,838.0,26.7,21.7,84.0,19.45,36.675182,6.0721655,11.976112,1019.067,5
2025-01-23 06:00:00,32.7,44.9,711.0,17.0,20.8,109.0,22.0,29.320587,5.6521144,9.16228,1019.07404,6
2025-01-23 07:00:00,21.7,37.3,567.0,11.1,17.7,120.0,24.2,23.996792,6.8423676,26.564985,1017.8811,7
2025-01-23 08:00:00,18.5,34.0,407.0,6.8,13.4,124.0,25.55,19.180855,6.889557,19.855309,1016.586,8
2025-01-23 09:00:00,15.8,29.7,302.0,4.3,10.1,126.0,26.1,16.046677,6.9270773,24.567156,1015.4881,9
2025-01-23 10:00:00,13.2,25.3,273.0,3.3,8.7,127.0,26.4,12.514948,8.415842,48.468323,1014.6897,10
2025-01-23 11:00:00,11.7,22.2,299.0,4.1,8.2,125.0,26.2,13.348256,10.094454,58.861095,1014.48926,11
2025-01-23 12:00:00,10.0,15.9,399.0,7.5,7.9,117.0,25.7,14.48816,9.82114,63.904633,1014.68774,12
2025-01-23 13:00:00,12.4,16.2,647.0,16.4,8.6,102.0,24.75,15.273955,12.031756,51.072456,1015.38464,13
2025-01-23 14:00:00,16.6,19.0,967.0,28.4,9.5,80.0,23.4,16.687405,10.152064,52.92685,1016.0804,14
```

This csv was also created just for the understanding and visualisation purpose and was not used anywhere.

6. AQI Calculation Using EPA Standard

CHALLENGE:

During the dataset inspection, I discovered that the API provided only pollutant concentration values (PM2.5, PM10, NO_2, CO, SO_2, O_3) and not a precomputed AQI column.

Since AQI was the main forecasting target, the absence of this variable made the dataset incomplete for supervised learning.

To solve this issue:

- I implemented a custom AQI calculation module based on the official standard defined by the United States Environmental Protection Agency
- Created breakpoint tables for each pollutant
- Applied EPA truncation rules
- Computed sub-indices using linear interpolation
- Selected the maximum sub-index as the final AQI
- Identified the dominant pollutant

```
src > utils > aqi_calculator.py > ...
import numpy as np

# ================ BREAKPOINT TABLES ================

PM25_BREAKPOINTS = [
    (0.0, 12.0, 0, 50),
    (12.1, 35.4, 51, 100),
    (35.5, 55.4, 101, 150),
    (55.5, 150.4, 151, 200),
    (150.5, 250.4, 201, 300),
    (250.5, 350.4, 301, 400),
    (350.5, 500.4, 401, 500),
]

PM10_BREAKPOINTS = [
    (0, 54, 0, 50),
    (55, 154, 51, 100),
    (155, 254, 101, 150),
    (255, 354, 151, 200),
    (355, 424, 201, 300),
    (425, 504, 301, 400),
    (505, 604, 401, 500),
]

CO_BREAKPOINTS = [
    (0.0, 4.4, 0, 50),
    (4.5, 9.4, 51, 100),
    (9.5, 12.4, 101, 150),
    (12.5, 15.4, 151, 200),
    (15.5, 30.4, 201, 300),
    (30.5, 40.4, 301, 400),
```

```
rc > utils > aqi_calculator.py > ...

class EPAAQICalculator:
    def calculate_aqi(self, pm25=None, pm10=None, co=None, no2=None, o3=None, so2=None):
        sub_indexes = {}

        if pm25 is not None:
            pm25 = truncate(pm25, 1)
            sub_indexes["PM2.5"] = compute_sub_aqi(pm25, PM25_BREAKPOINTS)

        if pm10 is not None:
            pm10 = truncate(pm10, 0)
            sub_indexes["PM10"] = compute_sub_aqi(pm10, PM10_BREAKPOINTS)

        if co is not None:
            co = truncate(co, 1)
            sub_indexes["CO"] = compute_sub_aqi(co, CO_BREAKPOINTS)

        if no2 is not None:
            no2 = truncate(no2, 0)
            sub_indexes["NO2"] = compute_sub_aqi(no2, NO2_BREAKPOINTS)

        if o3 is not None:
            o3 = truncate(o3, 0)
            sub_indexes["O3"] = compute_sub_aqi(o3, O3_BREAKPOINTS)

        if so2 is not None:
            so2 = truncate(so2, 0)
            sub_indexes["SO2"] = compute_sub_aqi(so2, SO2_BREAKPOINTS)

        sub_indexes = {k: v for k, v in sub_indexes.items() if v is not None}

        if not sub_indexes:
            return np.nan, None

        dominant = max(sub_indexes, key=sub_indexes.get)
        return sub_indexes[dominant], dominant
```

After implementing this module, the AQI column was successfully generated and validated, making the dataset suitable for EDA, feature engineering and model training.

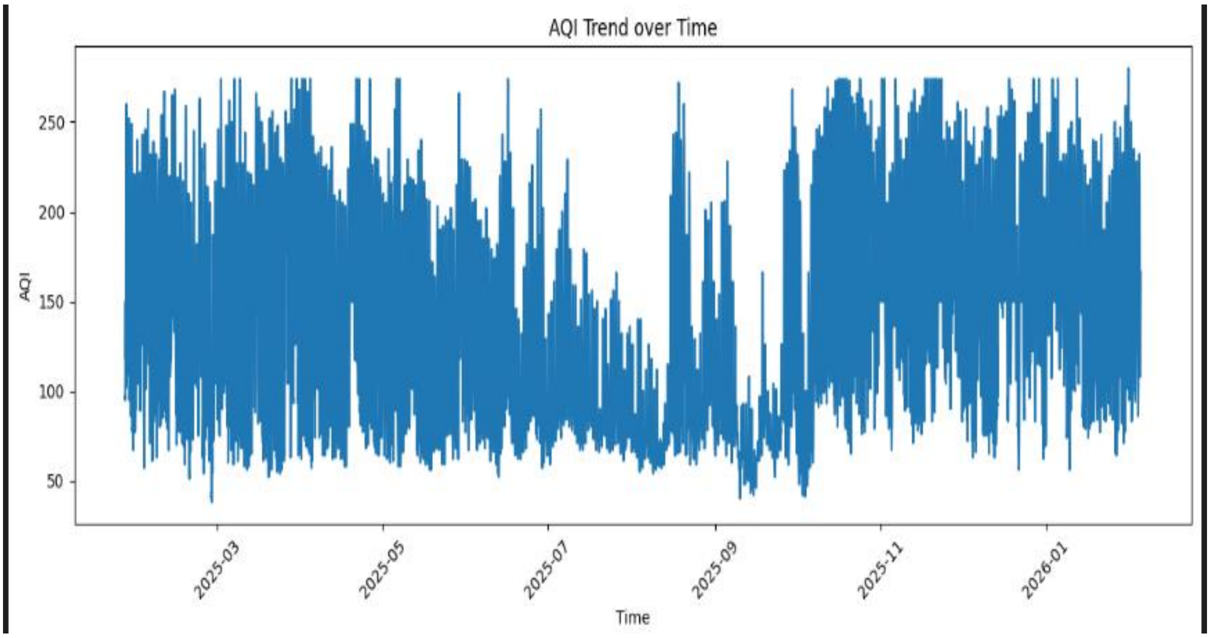
7. Exploratory Data Analysis (EDA)

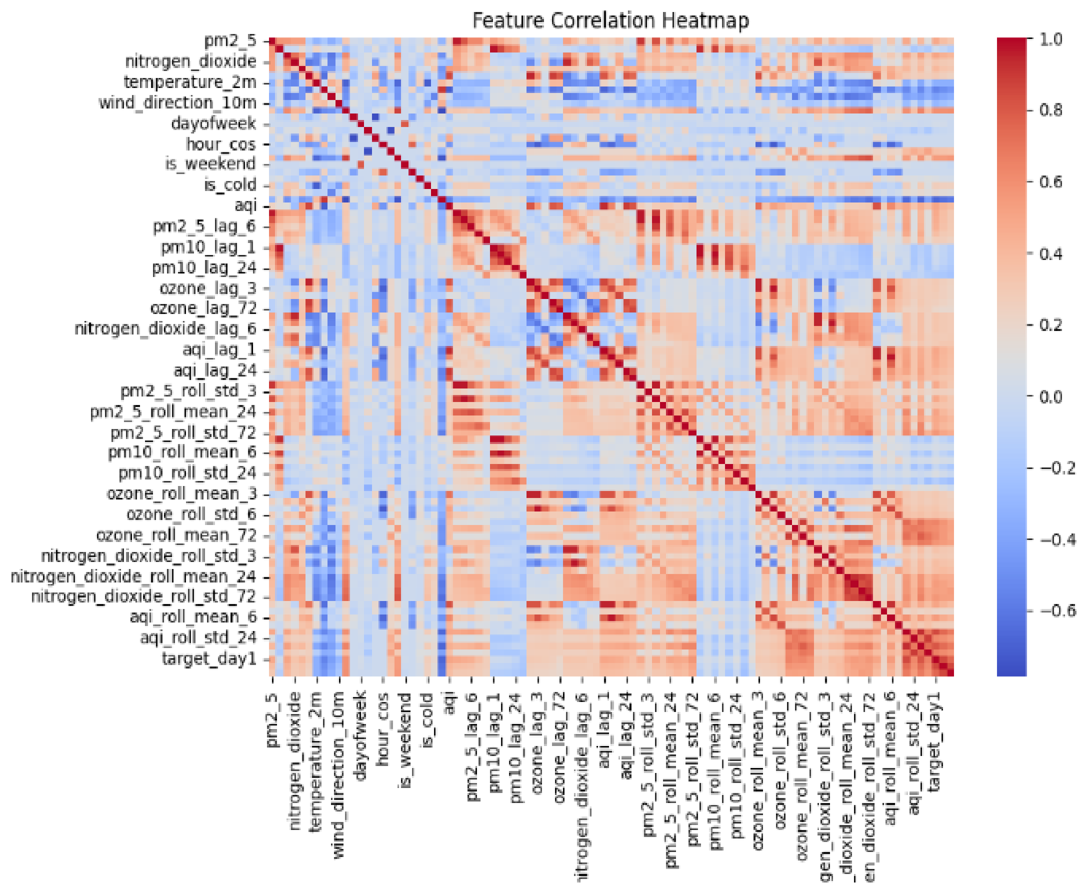
I also performed EDA on features to understand:

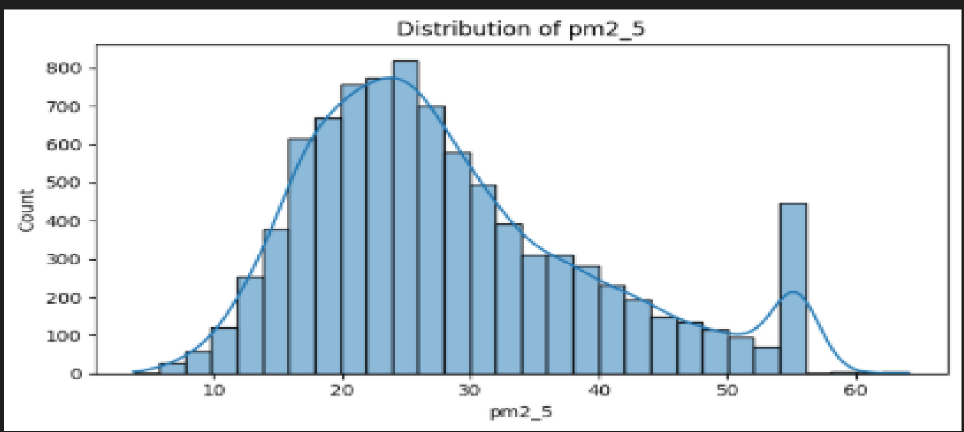
- Feature distributions
- Correlation between pollutants and AQI
- Seasonal patterns
- Trend behavior
- Relationship between weather and pollutants

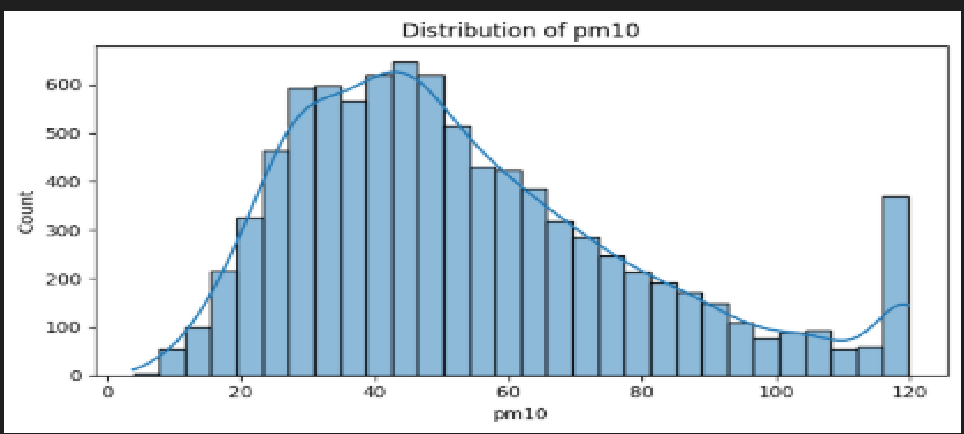
```
☁ Connecting to Hopsworks Feature Store...
2026-02-04 20:15:31,303 INFO: Initializing external client
2026-02-04 20:15:31,303 INFO: Base URL: https://c.app.hopsworks.ai:443
2026-02-04 20:15:34,051 INFO: Python Engine initialized.

Logged in to project, explore it here https://c.app.hopsworks.ai:443/p/1357975
Finished: Reading data from Hopsworks, using Hopsworks Feature Query Service (23.26s)
✅ Loaded 8997 rows from Feature Store
```

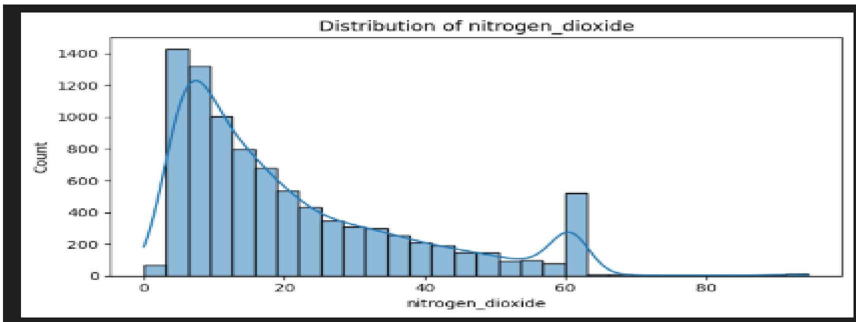
	timestamp	pm2_5	pm10	carbon_monoxide	nitrogen_dioxide	sulphur_dioxide	ozone	temperature_2m	relative_humidity_2m	wind_speed_10m	...	aqi_roll_std_3	aqi_roll_mean_6	aqi_roll_std_6
0	2025-04-22 08:00:00+00:00	36.8	119.8	505.0	6.4	27.7	175.5	35.2	40.700848	11.935778	...	24.172988	186.166667	71.182629
1	2025-07-28 00:00:00+00:00	18.6	52.5	137.0	5.6	7.7	58.0	27.9	81.797780	15.304234	...	4.582576	74.500000	4.415880
2	2025-06-18 21:00:00+00:00	20.5	43.2	139.0	6.1	7.6	59.0	28.8	85.630870	16.956345	...	1.527525	71.333333	1.505545
[illegible]	2025-06-01	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

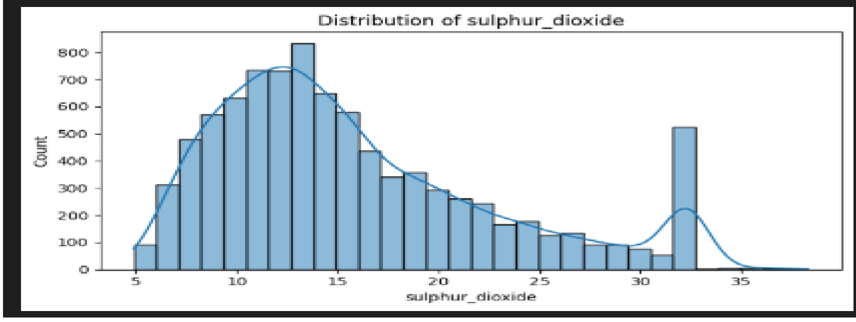


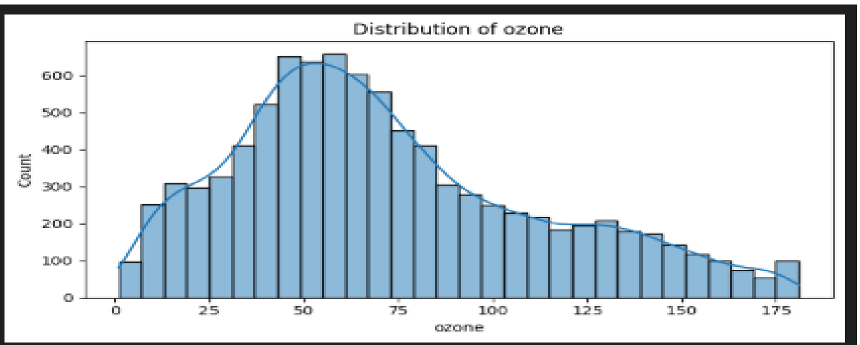

Feature Correlation Heatmap
pm2_5
nitrogen_dioxide
temperature_2m
wind_direction_10m
dayofweek
hour_cos
is_weekend
is_cold
aqi
pm2_5_lag_6
pm10_lag_1
pm10_lag_24
ozone_lag_3
ozone_lag_72
nitrogen_dioxide_lag_6
aqi_lag_1
aqi_lag_24
pm2_5_roll_std_3
pm2_5_roll_mean_24
pm2_5_roll_std_72
pm10_roll_mean_6
pm10_roll_std_24
ozone_roll_mean_3
ozone_roll_std_6
ozone_roll_mean_72
nitrogen_dioxide_roll_std_3
nitrogen_dioxide_roll_mean_24
nitrogen_dioxide_roll_std_72
aqi_roll_mean_6
aqi_roll_std_24
target_day1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

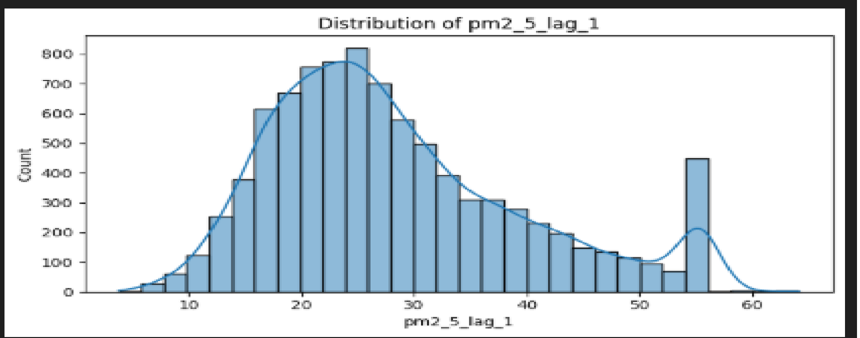

Distribution of pm2_5
Count
0
100
200
300
400
500
600
700
800
10
20
30
40
50
60
pm2_5


Distribution of pm10
Count
0
100
200
300
400
500
600
0
20
40
60
80
100
120
pm10

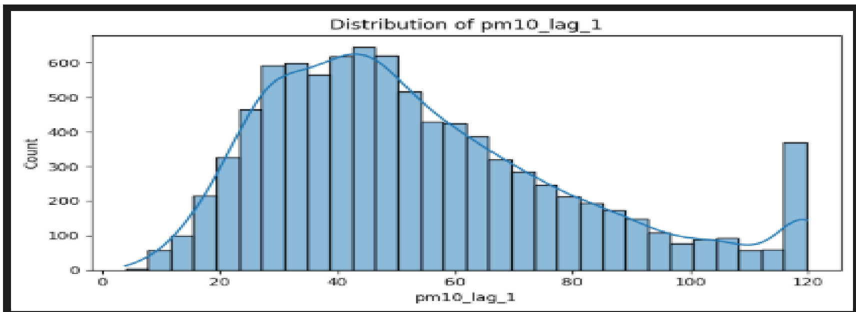

Distribution of nitrogen_dioxide
Count
nitrogen_dioxide

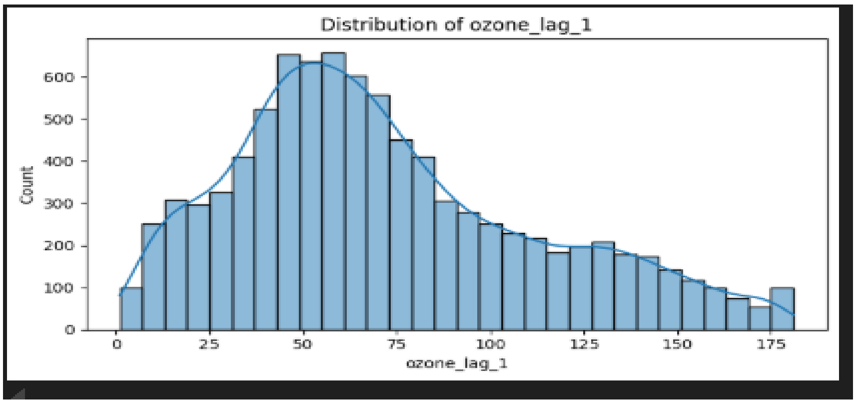

Distribution of sulphur_dioxide
Count
sulphur_dioxide

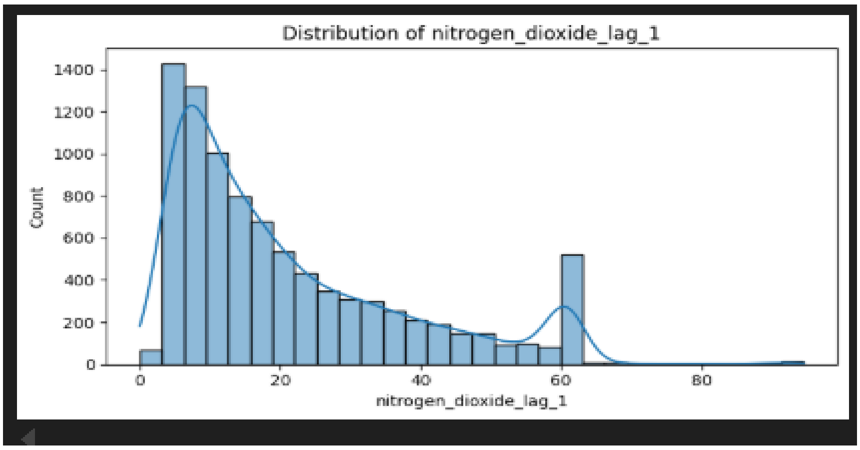

Distribution of ozone
Count
ozone


Distribution of pm2_5_lag_1
Count
pm2_5_lag_1


Distribution of pm2_5_lag_3


Distribution of pm10_lag_1
Count
pm10_lag_1





The detailed visualizations are shown in the dashboard and ipynb file.

Key observations:

• PM2.5 and PM10 showed strong correlation with AQI
• Wind speed negatively correlated with pollutant concentration
• Temperature had moderate impact on AQI trends
• AQI displayed clear temporal patterns

EDA helped guide feature engineering decisions.

8. Hopsworks Integration

I also implemented a Hopsworks connection test module to verify:

• Project authentication
• API key validity
• Feature group accessibility
• Proper feature insertion

The purpose of this step was to proactively validate the Hopsworks integration and prevent potential connection or configuration errors during future pipeline executions.

```
Logged in to project, explore it here https://c.app.hopswor
✅ Connected to project: AQIKarachi_predictor

✅ Feature Store ready
✅ Model Registry ready

============================================================
🎉 HOPSWORKS IS READY!
============================================================
2026-02-02 20:19:28,734 INFO: Closing external client and c
Connection closed.
```

9. Feature Engineering and Feature Pipeline

Then , I implemented a feature pipeline that initially created a **feature group** where data for **365 days** was fetched and stored after all the preprocessing in the feature store. After the initial setup, I changed the pipeline configuration to fetch only the latest **7 days** of data for regular updates.

The data was collected using the **DataFetcher class** from the data_fetcher file to retrieve air quality data along with weather data from APIs. I also used the **EPA AQICalculator class** from the AQI calculation file to compute standardized AQI values based on pollutant concentrations.

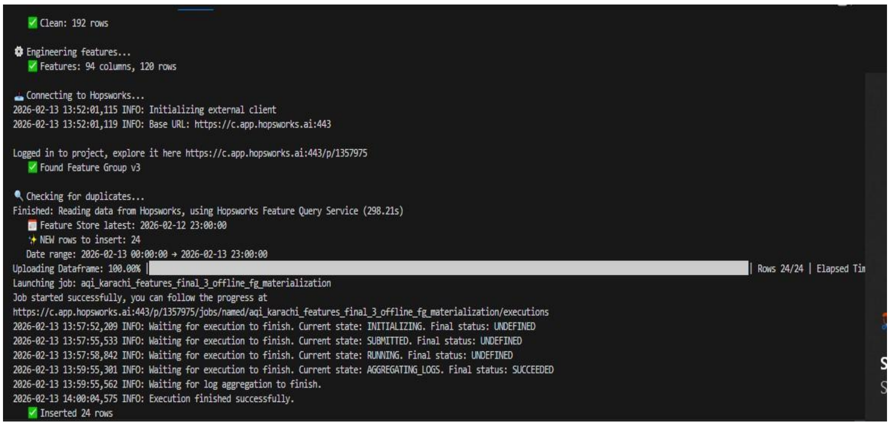
For preprocessing, I used the **DataCleaner class** from the data_cleaning file. In this step, unnecessary columns were removed, missing values were handled and outliers were capped to ensure clean and consistent data.

After cleaning, **feature engineering** was applied to generate useful features for model training. The processed data was then inserted into the feature store. The pipeline is designed in a way that **duplicates are not inserted**; only new unique rows are incrementally added during each run.

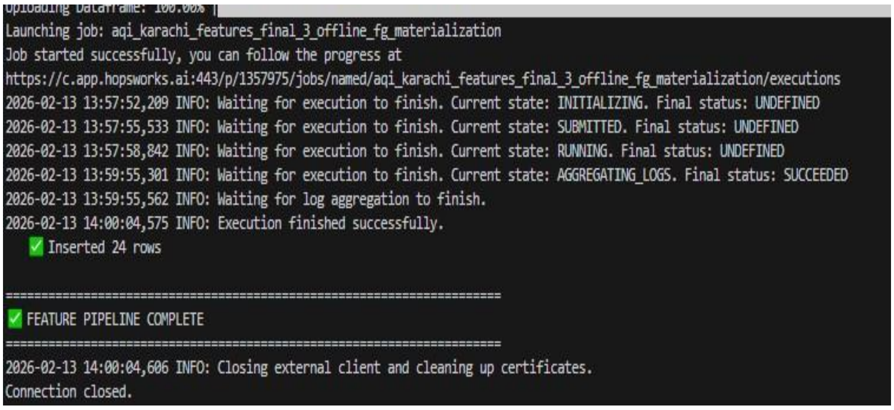
Overall, the pipeline automates data fetching, cleaning, feature engineering, and feature storage while maintaining updated and duplicate-free data.

From execution:

```
======================================================================
🚀 Feature Pipeline - 2026-02-13 13:50:40.005188
======================================================================

📥 Fetching 7 days from API...
📥 Fetching AIR QUALITY from 2026-02-06 to 2026-02-13
✅ Air-quality rows: 192
🌦Fetching WEATHER from 2026-02-06 to 2026-02-13
✅ Weather rows: 192
🔗 Merging air-quality + weather data...
🎯 Final merged rows: 192
📅 Date range: 2026-02-06 00:00:00 to 2026-02-13 23:00:00
   ✅ Fetched: 192 rows

🧹 Cleaning data...
🧹 Starting data cleaning...
⚠Dropping empty columns: ['ammonia']
⚠Capping 10 outliers in pm2_5
⚠Capping 7 outliers in pm10
⚠Capping 8 outliers in carbon_monoxide
⚠Capping 1 outliers in nitrogen_dioxide
⚠Capping 9 outliers in sulphur_dioxide
✅ Data cleaning complete
   ✅ Clean: 192 rows
```



```
✅ Clean: 192 rows

⚙ Engineering features...
  ✅ Features: 94 columns, 120 rows

Connecting to Hopsworks...
2026-02-13 13:52:01,115 INFO: Initializing external client
2026-02-13 13:52:01,119 INFO: Base URL: https://c.app.hopsworks.ai:443

Logged in to project, explore it here https://c.app.hopsworks.ai:443/p/1357975
  ✅ Found Feature Group v3

🔍 Checking for duplicates...
Finished: Reading data from Hopsworks, using Hopsworks Feature Query Service (298.21s)
  📅 Feature Store latest: 2026-02-12 23:00:00
  ✨ NEW rows to insert: 24
  Date range: 2026-02-13 00:00:00 → 2026-02-13 23:00:00
Uploading Dataframe: 100.00% |                    | Rows 24/24 | Elapsed Tim
Launching job: aqi_karachi_features_final_3_offline_fg_materialization
Job started successfully, you can follow the progress at
https://c.app.hopsworks.ai:443/p/1357975/jobs/named/aqi_karachi_features_final_3_offline_fg_materialization/executions
2026-02-13 13:57:52,209 INFO: Waiting for execution to finish. Current state: INITIALIZING. Final status: UNDEFINED
2026-02-13 13:57:55,533 INFO: Waiting for execution to finish. Current state: SUBMITTED. Final status: UNDEFINED
2026-02-13 13:57:58,842 INFO: Waiting for execution to finish. Current state: RUNNING. Final status: UNDEFINED
2026-02-13 13:59:55,301 INFO: Waiting for execution to finish. Current state: AGGREGATING_LOGS. Final status: SUCCEEDED
2026-02-13 13:59:55,562 INFO: Waiting for log aggregation to finish.
2026-02-13 14:00:04,575 INFO: Execution finished successfully.
  ✅ Inserted 24 rows
```



```
Launching job: aqi_karachi_features_final_3_offline_fg_materialization
Job started successfully, you can follow the progress at
https://c.app.hopsworks.ai:443/p/1357975/jobs/named/aqi_karachi_features_final_3_offline_fg_materialization/executions
2026-02-13 13:57:52,209 INFO: Waiting for execution to finish. Current state: INITIALIZING. Final status: UNDEFINED
2026-02-13 13:57:55,533 INFO: Waiting for execution to finish. Current state: SUBMITTED. Final status: UNDEFINED
2026-02-13 13:57:58,842 INFO: Waiting for execution to finish. Current state: RUNNING. Final status: UNDEFINED
2026-02-13 13:59:55,301 INFO: Waiting for execution to finish. Current state: AGGREGATING_LOGS. Final status: SUCCEEDED
2026-02-13 13:59:55,562 INFO: Waiting for log aggregation to finish.
2026-02-13 14:00:04,575 INFO: Execution finished successfully.
  ✅ Inserted 24 rows

======================================================================
✅ FEATURE PIPELINE COMPLETE
======================================================================
2026-02-13 14:00:04,606 INFO: Closing external client and cleaning up certificates.
Connection closed.
```

10. Model Training Pipeline

After this, I implemented a training pipeline that:

- Pulls features from Hopsworks
- Splits data correctly (time-series aware split)
- Trains three models (Ridge,Randomforest,XGboost)
- Evaluates model performance using three evaluation metrics: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2) to measure prediction accuracy and reliability
- Shows best model for each day based on the lowest RMSE
- Register all the models in the Model Registry

CHALLENGE:

During model training, I identified a data leakage problem where future information was indirectly influencing the training process.

Initially, the model was producing unusually high R^2 scores, which indicated that it might be accessing information from the future. The performance appeared too good to be realistic for a real-world AQI forecasting problem.

After investigation, I corrected the issue by:

- Ensuring proper time-based splits
- Removing any future-dependent features
- Validating chronological order

After fixing the leakage, the R^2 scores reduced to approximately 0.6–0.76 Although the performance decreased compared to the leaked setup, the results became realistic, reliable and suitable for real-world deployment.

This step significantly improved the credibility and robustness my forecasting system.

Model Performance Results

```
2026-02-14 18:27:00,696 INFO: Initializing external client
2026-02-14 18:27:00,697 INFO: Base URL: https://c.app.hopsworks.ai:443
2026-02-14 18:27:09,219 INFO: Python Engine initialized.

Logged in to project, explore it here https://c.app.hopsworks.ai:443/p/1357975
======================================================================
TRAINING PIPELINE - 2026-02-14 18:27:12.986985
======================================================================
Finished: Reading data from Hopsworks, using Hopsworks Feature Query Service (19.88s)
Loaded 9237 rows from Feature Store
Train samples: 7332
Test samples:  1833

TRAINING


TARGET: DAY1
Ridge        | RMSE:  31.12 | MAE:  24.85 | R2: 0.698 | OK
```

```
TRAINING


TARGET: DAY1
Ridge        | RMSE:  31.12 | MAE:  24.85 | R2: 0.698 | OK
RandomForest | RMSE:  28.69 | MAE:  22.81 | R2: 0.743 | OK
XGBoost      | RMSE:  29.55 | MAE:  23.58 | R2: 0.728 | OVERFITTING

TARGET: DAY2
Ridge        | RMSE:  33.30 | MAE:  26.94 | R2: 0.657 | OK
RandomForest | RMSE:  30.49 | MAE:  24.86 | R2: 0.713 | OK
XGBoost      | RMSE:  30.33 | MAE:  24.45 | R2: 0.716 | OK

TARGET: DAY3
Ridge        | RMSE:  35.31 | MAE:  28.67 | R2: 0.617 | OK
RandomForest | RMSE:  30.02 | MAE:  24.40 | R2: 0.723 | OK
XGBoost      | RMSE:  30.59 | MAE:  24.38 | R2: 0.712 | OK

BEST MODEL PER HORIZON
```

Showing best model for each day based on the lowest RMSE metric:

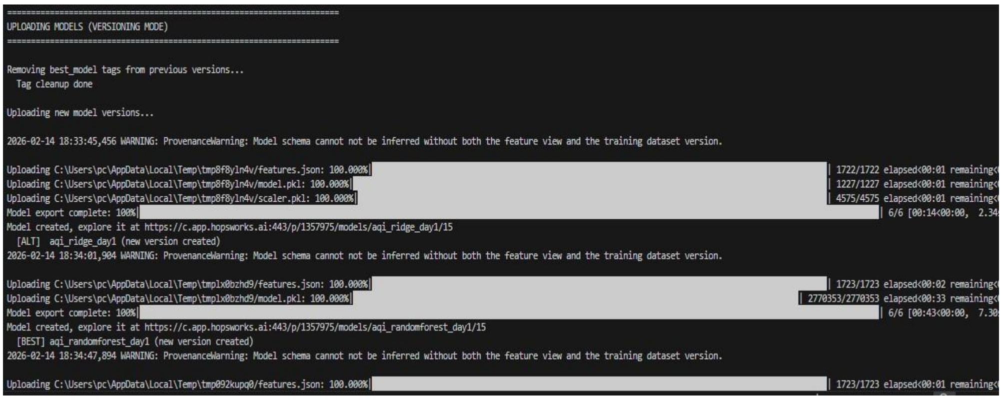
```
BEST MODEL PER HORIZON

DAY1 -> RandomForest (RMSE = 28.69)
DAY2 -> XGBoost (RMSE = 30.33)
DAY3 -> RandomForest (RMSE = 30.02)

TRAINING DONE
```

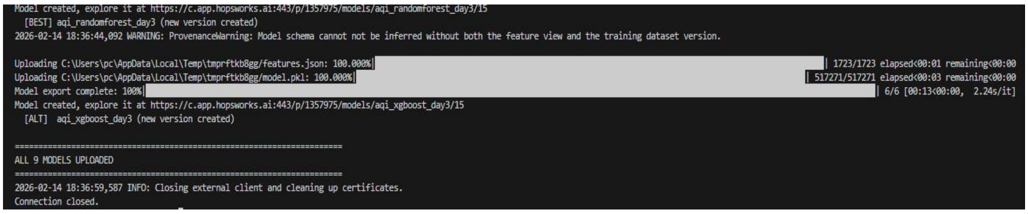
11. Model Registry and Versioning

Each trained model was :

• Uploaded to Hopsworks Model Registry
• Versioned properly
• Tagged as BEST and ALT model per day
• Stored with metadata

From execution:





All 9 models were uploaded successfully.

This allows:

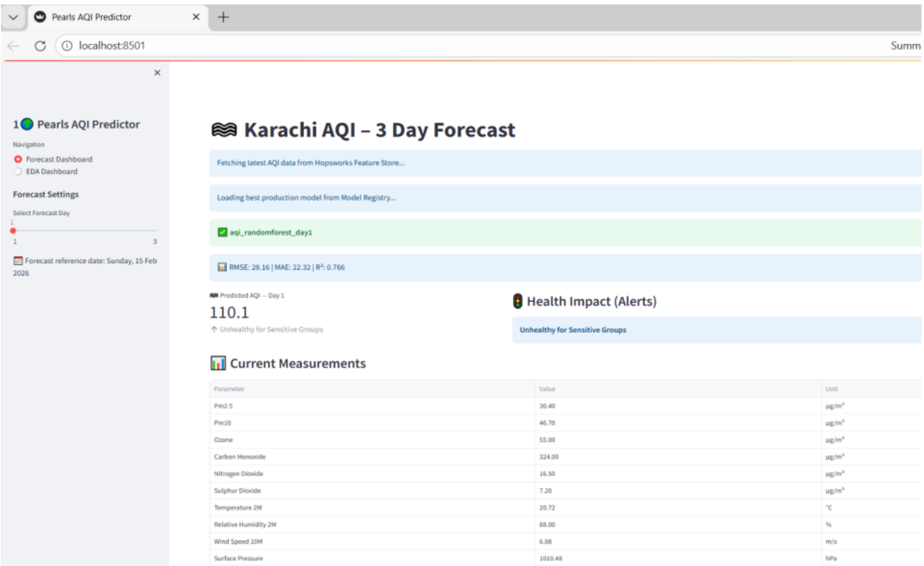
- Model reproducibility
- Performance tracking
- Rollback capability

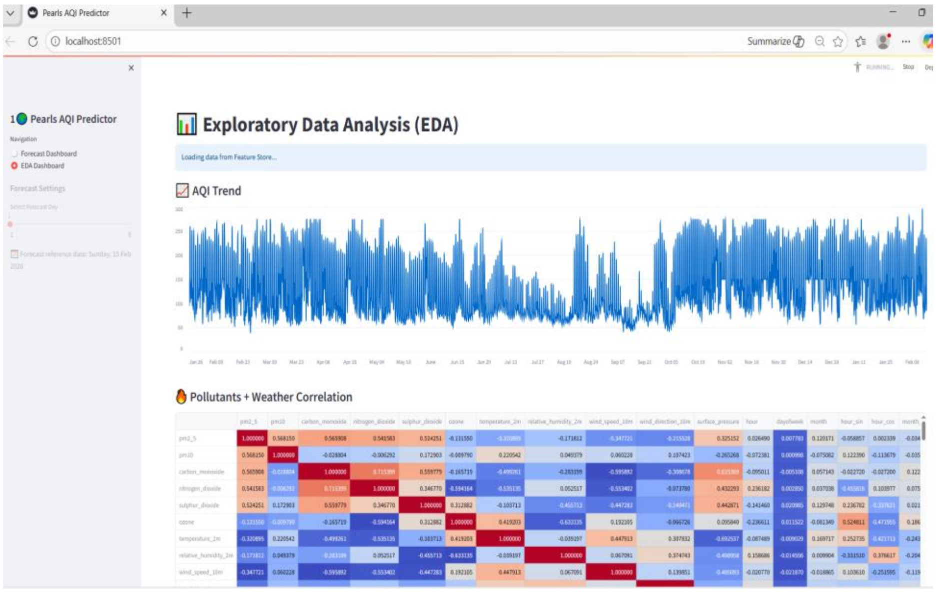
12. Streamlit Deployment

The final best models were then deployed.

12.1 Local Deployment

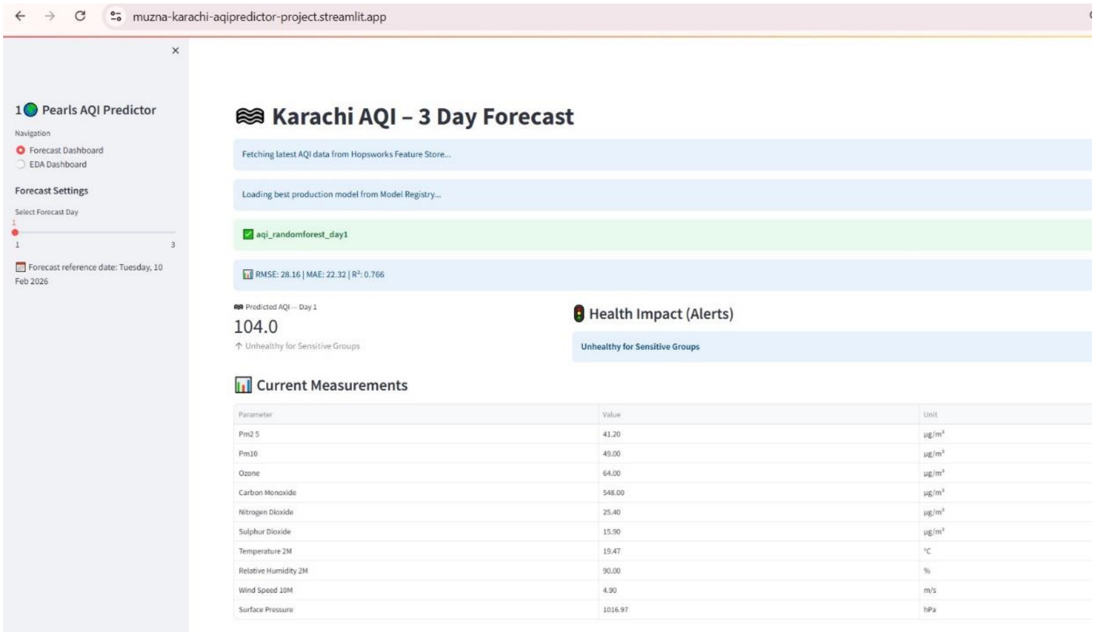
- Using Streamlit on localhost
- Tested predictions
- Verified API integration

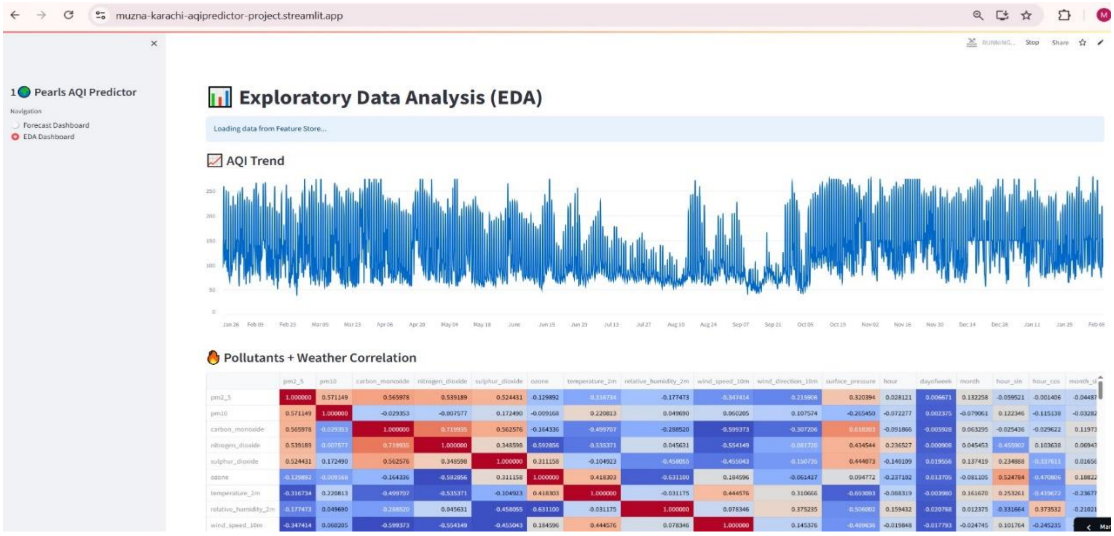




Then, I deployed it on streamlit cloud.

12.2 Cloud Deployment





The deployed application:

• Displays forecast and Eda dashboard

• Fetches latest feature data

• Loads latest best model

• Generates AQI forecasts

• Displays forecast predictions interactively

•Displays Eda visualizations

13. SHAP Explainability

To ensure model interpretability, I also implemented SHAP (SHapley Additive Explanations).

SHAP helped:

• Identify most important features

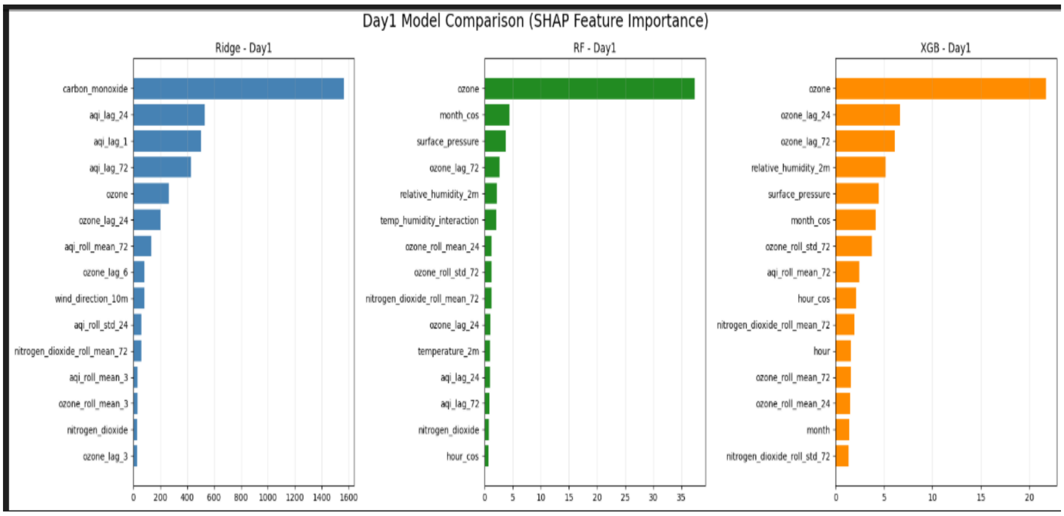
• Understand feature impact direction

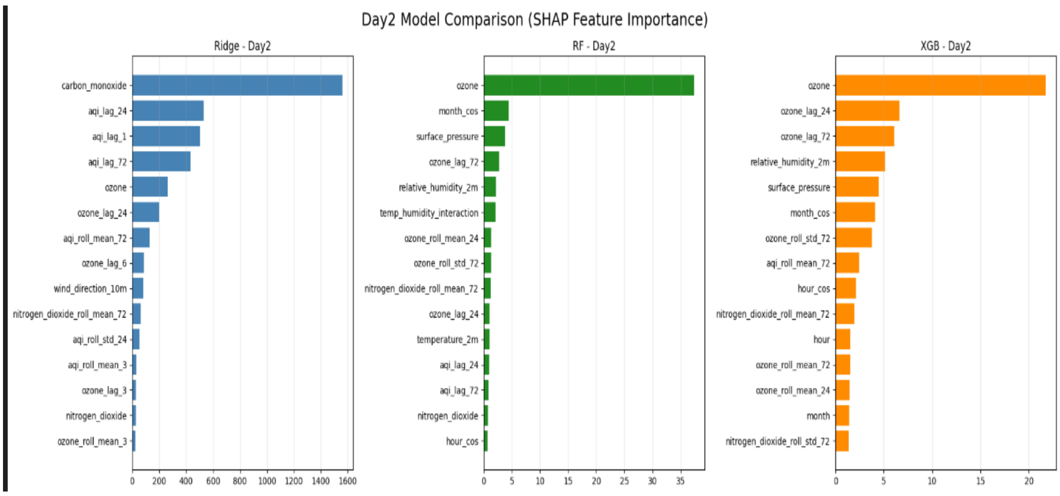
• Validate model behavior

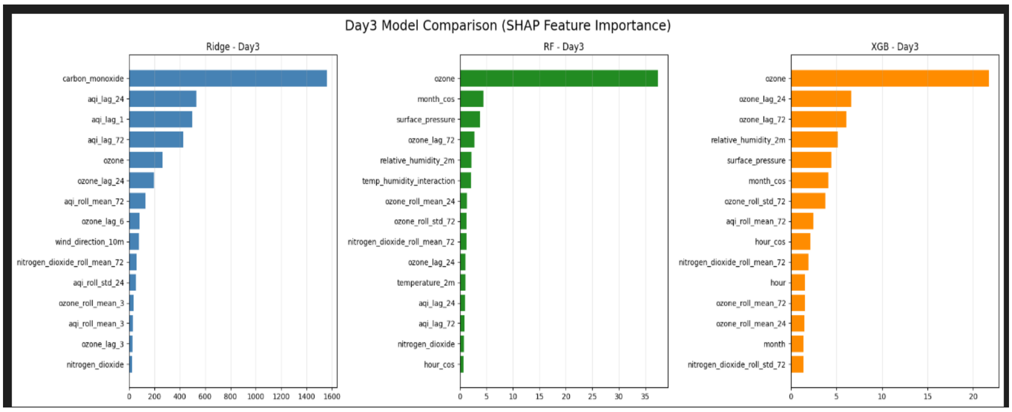
• Increase trust in predictions

This made the system both predictive and explainable.

```
Logged in to project, explore it here https://c.app.hopsworks.ai:443/p/1357975
Finished: Reading data from Hopsworks, using Hopsworks Feature Query Service (23.82s)
✅ SHAP sample ready: (500, 89)

Downloading: 0.000%|          | 0/4575 elapsed<00:00 remaining<?

Downloading model artifact (0 dirs, 1 files)...

Downloading: 0.000%|          | 0/1227 elapsed<00:00 remaining<?

Downloading model artifact (0 dirs, 2 files)...

Downloading: 0.000%|          | 0/1722 elapsed<00:00 remaining<?

Downloading model artifact (0 dirs, 3 files)... DONE

Downloading: 0.000%|          | 0/2666817 elapsed<00:00 remaining<?

Downloading model artifact (0 dirs, 1 files)...

Downloading: 0.000%|          | 0/1723 elapsed<00:00 remaining<?

Downloading model artifact (0 dirs, 2 files)... DONE
```







14. CI/CD Implementation (GitHub Actions)

At last, after pushing my project to github, I implemented CI/CD using GitHub Actions.

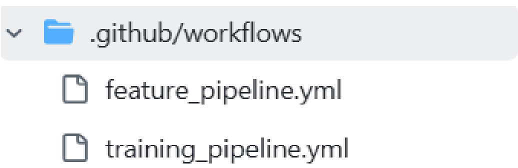
Two workflow files were created:

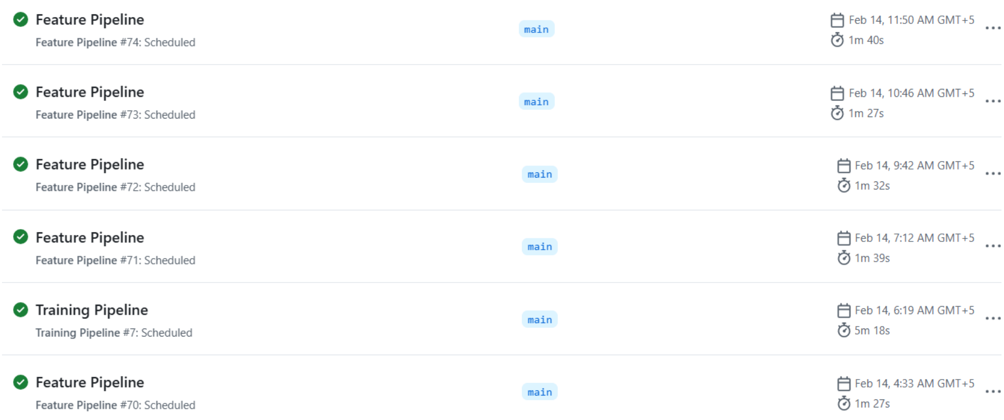
1) Feature Pipeline Workflow (feature_pipeline.yml)

- Scheduled hourly
- Fetches new data
- Processes features
- Updates Feature Store

2) Training Pipeline Workflow (training_pipeline.yml)

- Scheduled daily
- Pulls latest features
- Retrains models
- Updates Model Registry





15. Challenges Faced And Solved

1. API selection and rate limiting issues
2. Ensuring complete one-year historical data
3. Missing AQI Target Variable
4. Fixing data leakage in training

Each challenge was resolved systematically through debugging and validation.

16. Conclusion

This project successfully implements a production-ready AQI forecasting system using:

• Open-Meteo API
• Data validation and cleaning
• Feature engineering
• Hopsworks Feature Store
• Multi-model training
• Model Registry with versioning
• Streamlit deployment
• GitHub Actions CI/CD automation

The system is:

• Automated
• Scalable
• Reproducible
• Version-controlled
• Deployable

It demonstrates practical implementation of Machine Learning Operations (MLOps) in a real-world time-series forecasting problem.