# Pearls AQI Predictor

Submitted by: Syeda Anoosha Iqtidar (Data Science Domain)

**Live System:** https://hyderabad-pearls-aqi-predictor.streamlit.app/

## Introduction to the Project

Air pollution is a major public health issue in Hyderabad, Sindh. Emissions, industrial activity, weather inversions, and dust storms cause AQI levels to shift across hours and seasons.

Residents and officials lack reliable short-term forecasts. Most services show only current values or rely on simple assumptions. This project provides automated, explainable 72-hour hourly forecasts to support planning, public advisories, and health decisions.

## Problem Statement

The system predicts hourly AQI for the next 72 hours without future pollutant readings, relying on historical data and forecast weather since same-hour values can't be used. Leakage or poorly constructed rolling features could inflate metrics and cause failures production.

Time ordering remains strict; processing, splitting, and inference follow chronological order. The dataset is imbalanced, with 75% Moderate observations; severe events are rare. The system must operate continuously with hourly feature updates, daily retraining, low-latency serving, and failure handling without manual intervention.

The objective is to deliver accurate, explainable, and reliable 72-hour AQI forecasts under these constraints.

## Solution Offered

The system is a fully automated end-to-end AQI forecasting pipeline.

Hourly weather and air quality data are collected from Open-Meteo APIs. A Python feature pipeline generates 50+ time-aware features, strictly excluding same-hour pollutants to prevent leakage. Features are stored and versioned in Hopsworks.

Five models are retrained daily using chronological splits. The best model, usually LightGBM, is selected based on validation RMSE and registered automatically.

Forecasting uses an autoregressive 72-hour loop, updating lag features with predicted values at each step. Predictions are served through a FastAPI backend and visualized in a Streamlit dashboard with SHAP-based explanations and health alerts.

## System Working

The system operates in four stages:

Hourly Feature Pipeline triggers hourly via GitHub Actions, fetching 26 hours of weather and AQI data, merging by timestamp, engineering features, imputing causally, removing NaNs, and inserting into Hopsworks. It also processes forecast data for inference.

Daily Model Retraining runs daily, loading historical features, maintaining order, removing same-hour pollutants, encoding season, splitting data 90/10, training five models, evaluating RMSE, MAE, $R^2$, selecting and registering top models, and generating a 72-hour forecast snapshot.

Inference loads the best model when accessed, retrieves recent data and forecasts weather, then runs a 72-step autoregressive prediction loop, updating lag values each step.

User Interface displays AQI, forecasts, health advice, trends, alerts, and metrics via a fast, cached Streamlit dashboard.

# Feature Pipeline

## Data Collection

The pipeline uses three Open-Meteo endpoints.

The weather archive offers hourly data on temperature, humidity, pressure, wind, and precipitation for Hyderabad (25.3792°N, 68.3683°E). The air quality archive provides PM10, PM2.5, $NO_2$, $SO_2$, CO, and US AQI. The forecast API predicts these variables up to 4 days ahead for a 72-hour period and timezone offset.

All requests use the Asia/Karachi timezone through the openmeteo_requests client with caching and automatic retries.

## Feature Engineering

Twelve raw variables are expanded into 50+ model features through six steps.

Temporal features include hour, weekday, day, month, quarter, week of year, weekend flag, daytime flag, and season. Sine and cosine encodings are applied to hour, weekday, and month to preserve cyclic structure.

Lag features capture persistence. PM2.5 and CO are lagged at 1, 3, 6, 12, and 24 hours. Temperature and pressure receive 12-hour lags to reflect inversion cycles. Lag intervals were selected using cross-correlation with AQI and mutual information. A 48-hour lag was tested but removed due to limited gain.

AQI change-rate features encode momentum using a strictly causal shift. Four absolute change features and three clipped rate features are generated.

Interaction features include temperature × humidity and wind speed × pressure. These approximate stagnation and ventilation effects. Scaling is handled later, where required.

Data quality detection flags hours where key weather variables change by less than 1% of their range, indicating potential duplication. This flag is removed before training.

Missing value handling applies forward fill with short limits, followed by a backward-only 12-hour rolling median. Only past values are used to avoid leakage.

## Leakage Prevention at Training and Prediction Time

Same-hour pollutants, the duplication flag, and the PM2.5-to-PM10 ratio are removed before model input. Only lagged pollutant features are retained.

The season variable is one-hot encoded. The same preparation steps are applied in training and inference to prevent train-serve mismatch.

# Exploratory Data Analysis

## Data Characteristics

- **Scope**: The dataset contains 17,520 records and 14 features.
- **Variables**:
    - **Weather**: Temperature, relative humidity, pressure, wind speed, wind direction, and precipitation.
    - **Pollutants**: PM10, PM2.5, $NO_2$, $SO_2$, Ozone ($O_3$), and Carbon Monoxide (CO).
    - **Target**: US Air Quality Index (AQI).

## Data Quality

After preprocessing, missing values were below 0.5%.

Missing Values were handled using a combination of forward-filling (limit 2-3) and linear interpolation for weather data. Pollutants were filled using forward-filling and a 12-hour rolling median to preserve local trends.

A custom function was used to identify "duplicate weather patterns" (records where features changed by less than 1%), finding 204 instances.

Data was checked against physical boundaries (e.g., AQI 0-600, Temperature -40 to 60°C). The actual data showed temperatures ranging from 7.3°C to 49.0°C and AQI levels from 33.7 to 167.0.

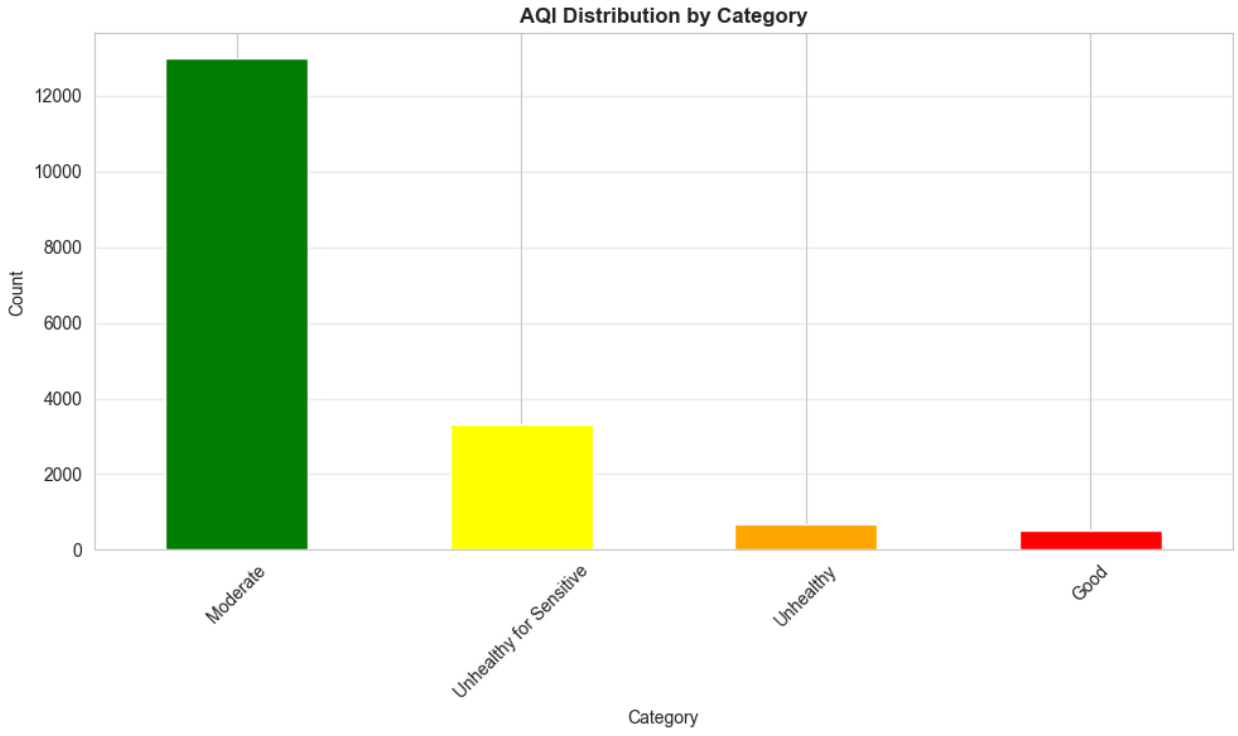
# AQI Distribution

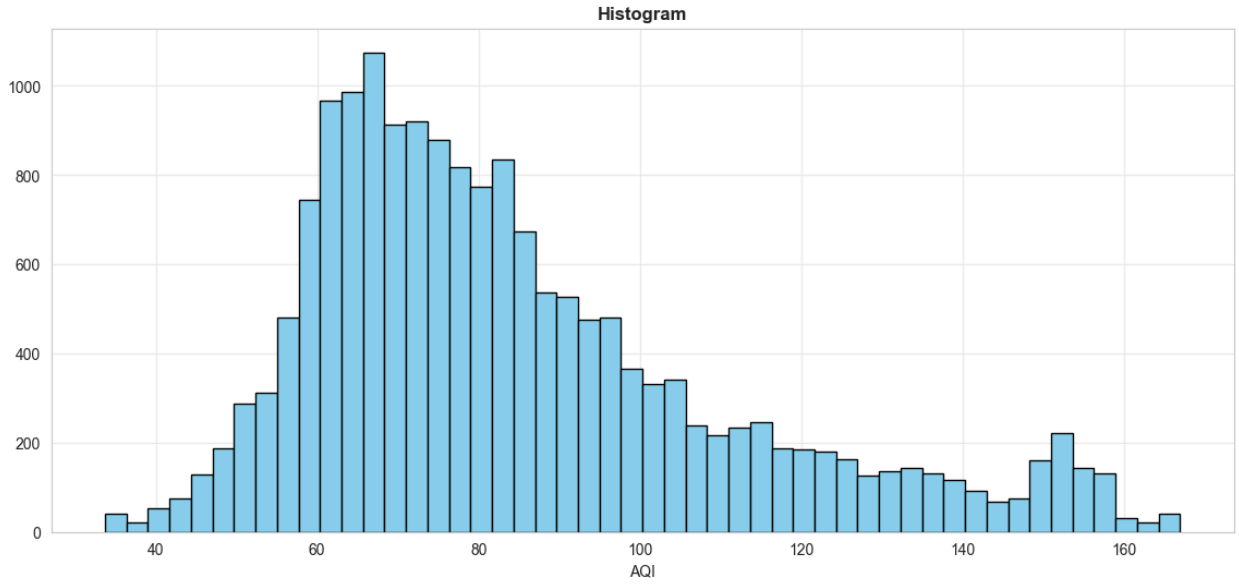
Mean AQI was 84.96, median 78.61, standard deviation 26.38, with values from 33.7 to 167.0. The distribution is right-skewed and concentrated between 60 and 110.

Category breakdown:

- Moderate: 74.18% (AQI 51-100)
- Unhealthy for Sensitive: 18.85% (AQI 101-150)
- Unhealthy: 3.98%(AQI 151-200)
- Good: 2.99% (AQI 0-50)
- Very Unhealthy or Hazardous <1% (AQI above 200)

This imbalance means the model sees mostly moderate conditions and limited extreme events. Average-case prediction is easier than spike prediction.

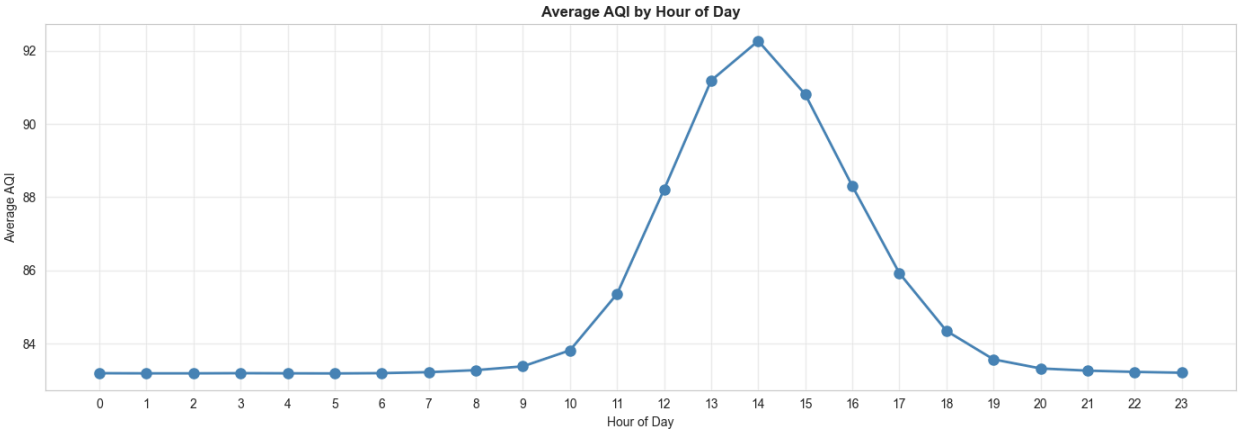





# Diurnal (Hourly) Pattern

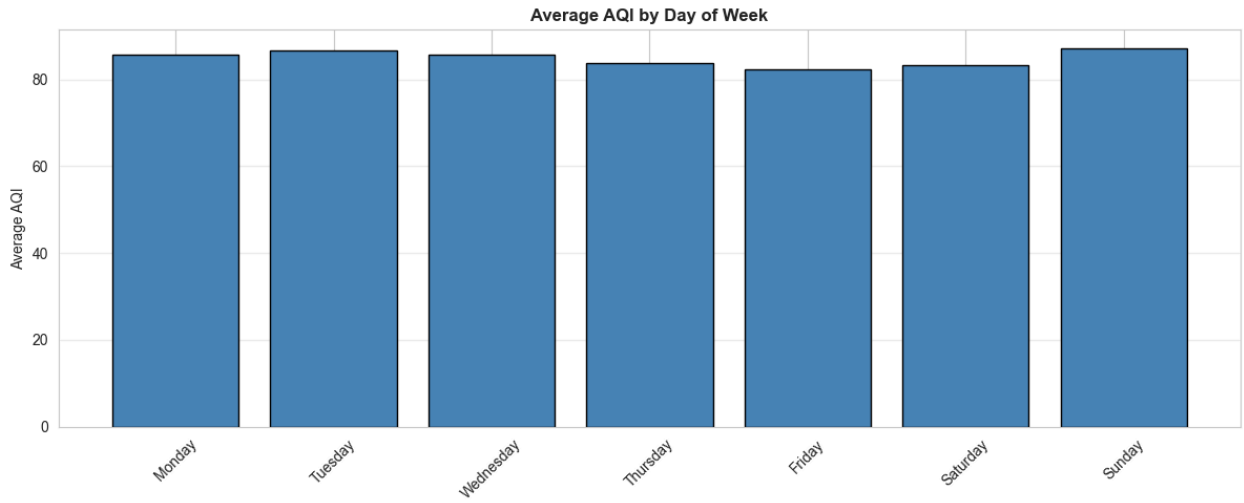
AQI in Hyderabad follows a clear daily cycle. Starting around 9 AM, AQI rises with traffic and solar heating, peaking between 1-3 PM when high temperatures trap pollutants near the ground. In the evening (4-8 PM), rush-hour traffic adds emissions, but cooling temperatures enhance mixing, gradually lowering AQI at night.

This pattern justified including hour-of-day and cyclic encodings. It also supports the 24-hour lag feature.




# Weekly Pattern

Weekly effect is minimal (~0.7 AQI difference). No strong weekday-weekend distinction, unlike some cities with sharp industrial shutdown effects. Day-of-week is retained as a feature for completeness, but contributes modestly to predictions.

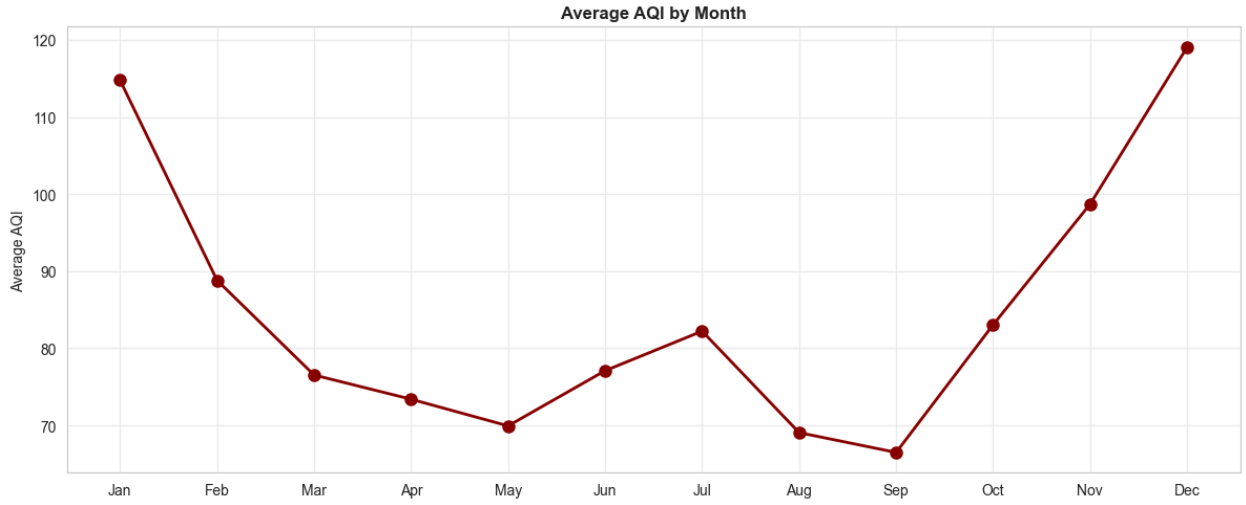


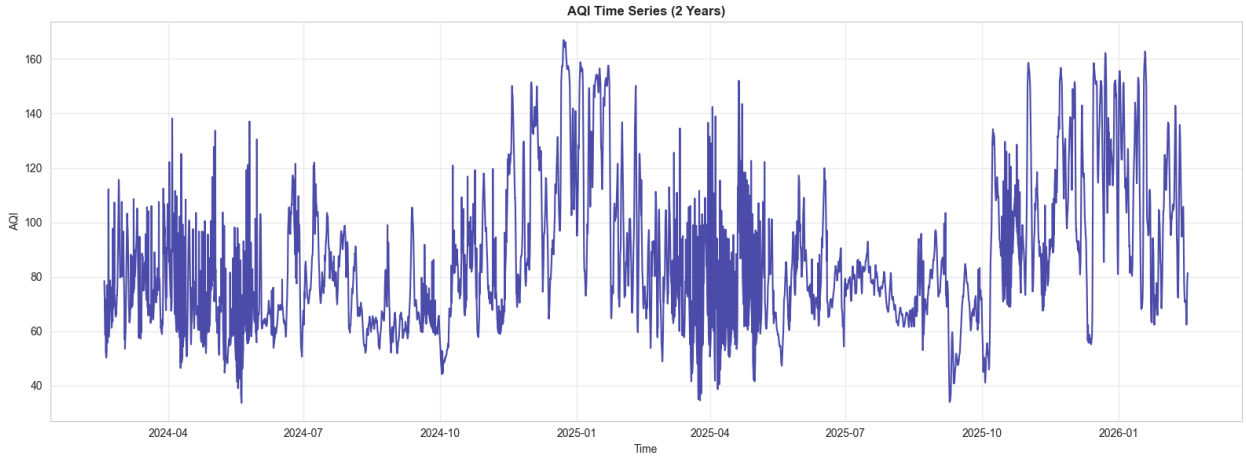

# Seasonal Pattern

Seasonal variation is stronger than hourly variation.

- **Winter Peak (November - February):** This is the most critical period. AQI averages climb significantly, peaking in December at nearly 120. This is likely due to "temperature inversion," where cold air traps pollutants near the ground, combined with increased heating requirements.
- **Spring/Summer Decline (March - June):** There is a steady improvement as temperatures rise. The air quality reaches a moderate baseline during these months.
- **Late Summer Dip (August - September):** The best air quality of the year occurs in September, with the average AQI dropping to its lowest point (around 65-67). This may be attributed to increased rainfall or wind patterns that help disperse particulate matter.
- **Autumn Surge (October - November):** As soon as October hits, there is a sharp vertical spike in pollution levels, returning the cycle to the hazardous winter levels.

The gap between winter peaks and fall lows confirms month and cyclic encoding as high-value features. It also supports daily retraining to handle seasonal drift.



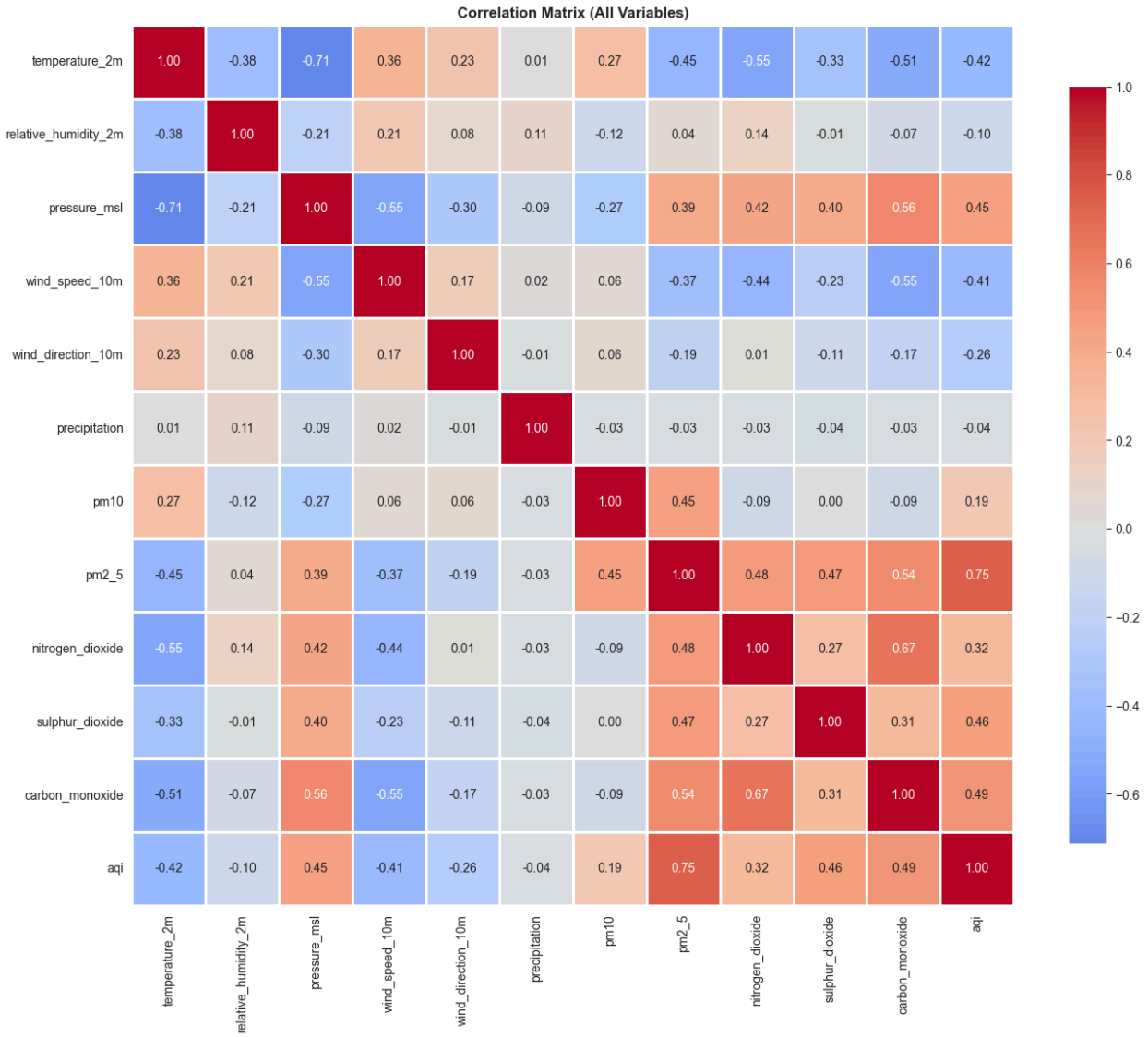

# Correlation Analysis

Pearson correlation identified PM2.5 as the strongest predictor at 0.75. This is expected: the US AQI formula is dominated by PM2.5 concentrations in most urban settings.

CO and $SO_2$ follow around 0.46 to 0.49, reflecting combustion-related emissions from traffic and industry. Pressure shows a positive correlation of 0.45. Temperature and wind speed are negatively correlated at roughly -0.42 and -0.41.

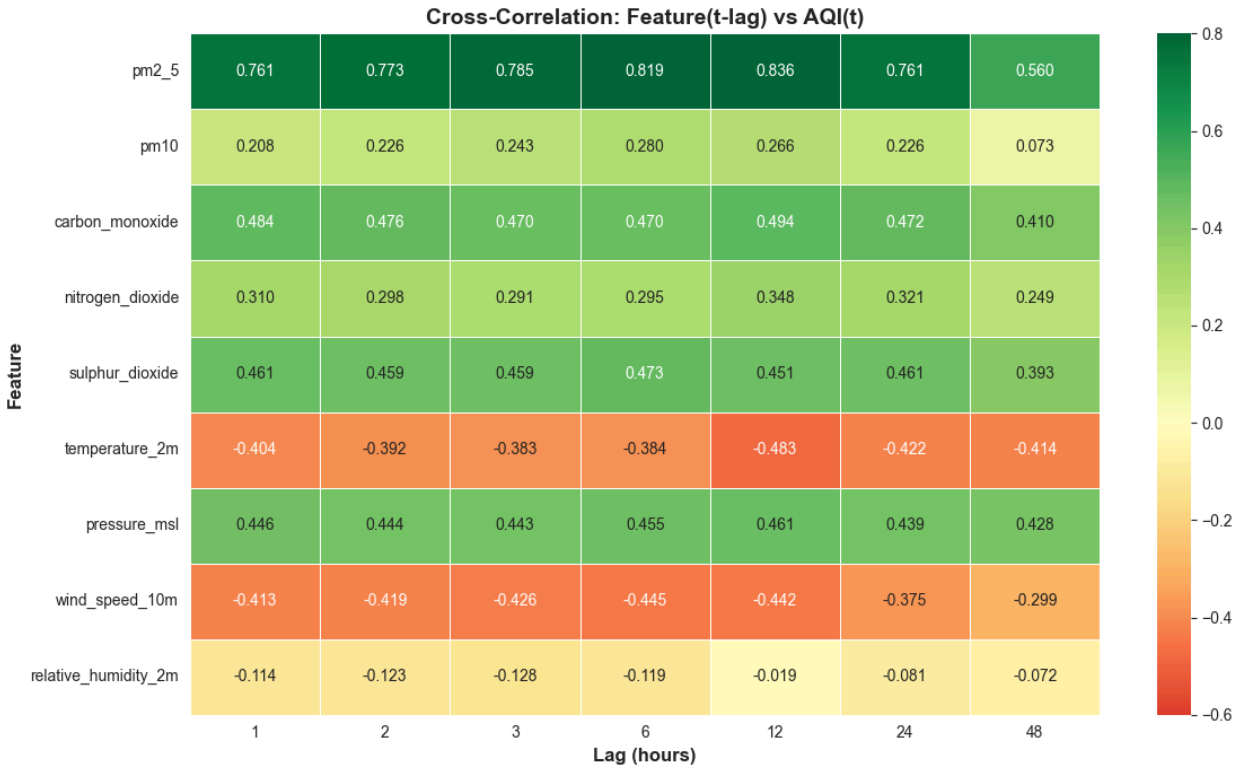
$NO_2$ and PM10 contribute modestly. Ozone, humidity, and precipitation show a weak linear association.

These findings drove lag selection and interaction features. PM2.5 and CO received multiple lags. Temperature, pressure, and wind speed informed interaction terms.




# Cross-Correlation: Feature Lags vs. AQI

PM2.5 dominates AQI prediction, correlations are high across all lags, peaking at 12h. Pressure and CO provide moderate predictive value, stable across lags. Temperature peaks at 12h (diurnal effect). Wind speed peaks at 6h, reflecting delayed pollutant dispersion.



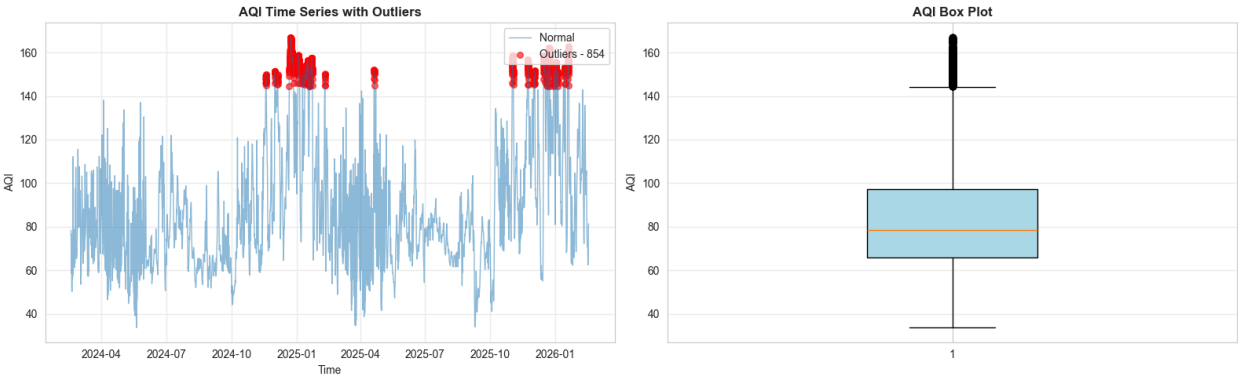

# Mutual Information Analysis

It confirms non-linear dependencies. Temperature and pressure are most informative at a 12h lag. PM2.5 is strong across both linear and non-linear metrics. CO and wind are weaker; no hidden relationships detected. Pressure shows a strong non-linear relationship (MI ~0.30). $NO_2$ and relative humidity have low MI, not useful.


# Outlier Assessment

Using IQR, 5% of observations were flagged, mostly above AQI 140. Z-score identified 0.25% above roughly 150.

These are real high-pollution events, not errors. Removing them would eliminate critical training examples. They were retained, with an underestimation of extreme spikes documented as a limitation.



## Key EDA Conclusions

PM2.5 is the dominant driver of AQI in Hyderabad. AQI shows a stable daily cycle and a strong seasonal pattern, with winter values several times higher than fall.

Lag features at 1h, 3h, and 24h capture persistence. A 12-hour lag captures inversion effects. Same-hour pollutants must be excluded to prevent leakage.

This justified lag intervals of 1h, 3h, 6h, 12h, and 24h for PM2.5 and CO, and 12h for temperature and pressure. The 48-hour lag added little value and was removed.

Class imbalance limits the model's exposure to extreme events, making rare spikes harder to predict accurately.

# Historical Data Backfill

A two-year historical dataset was collected to provide seasonal coverage before the hourly pipeline started. Data were retrieved in eight 90-day batches from the same Open-Meteo weather and air quality APIs.

Each batch was merged by timestamp, deduplicated, sorted, and processed with the same feature engineering as the hourly pipeline to maintain consistency. The final dataset contains roughly 17,500 feature rows (after removing lag-induced NaNs) and is stored in the Hopsworks Feature Store. The full backfill runs in 3-5 minutes on a standard runner.

# Model Training Pipeline

## Training Approach

The Model Training Pipeline trains and compares five models, LightGBM, XGBoost, Random Forest, ElasticNet, and a TensorFlow Neural Network, to predict AQI, selecting the best for deployment. Daily retraining updates the model with changing air quality patterns.

The training pipeline loads all historical features from Hopsworks, removes same-hour pollutants, one-hot encodes the season, and splits data chronologically: 10% for testing, remaining 90% into training and validation using scikit-learn's TimeSeriesSplit with 5 folds. No shuffling occurs.

All five models are trained and evaluated on train, validation, and test sets using RMSE, MAE, and $R^2$ metrics. Model selection ranks candidates by lowest validation RMSE, then MAE, and highest $R^2$ while filtering out overfitting models. Tree-based models outperform linear and neural networks, with lag features, especially the 24-hour PM2.5 lag, and temporal patterns like hour and month, being the most important predictors.

## Model Performance

| Model | Train RMSE | Train MAE | Train $R^2$ | Val RMSE | Val MAE | Val $R^2$ | Test RMSE | Test MAE | Test $R^2$ |
|---|---|---|---|---|---|---|---|---|---|
| Random Forest | 5.11 | 3.59 | 0.95 | 8.33 | 6.15 | 0.90 | 10.21 | 8.35 | 0.87 |
| XGBoost | 3.66 | 2.64 | 0.98 | 4.70 | 3.46 | 0.97 | 6.02 | 4.70 | 0.96 |
| LightGBM | 3.27 | 2.36 | 0.98 | 4.61 | 3.39 | 0.97 | 5.75 | 4.50 | 0.96 |
| ElasticNet | 6.76 | 5.04 | 0.92 | 6.62 | 5.03 | 0.94 | 8.43 | 6.83 | 0.91 |
| TensorFlow NN | 5.20 | 3.71 | 0.95 | 6.91 | 5.31 | 0.93 | 8.76 | 7.09 | 0.91 |

## Model Selection and Registration

Models with $R^2$=1.0 are excluded as overfit. Remaining models are ranked by validation RMSE; MAE and $R^2$ act as tiebreakers. All models are registered in Hopsworks for tracking and potential promotion.

## Feature Importance

Top features from LightGBM:

1. PM2.5 lag 24h
2. Hour of day
3. PM2.5 lag 1h
4. 24-hour AQI change rate
5. CO lag 24h
6. Temperature
7. Month
8. Pressure
9. PM2.5 lag 3h
10. Wind speed

Lag features dominate, confirming strong autocorrelation. Temporal and weather features capture diurnal/seasonal patterns and dispersion conditions.

## Autoregressive Inference

Forecasts are generated sequentially over 72 hours. After each hour, lag buffers for AQI, PM2.5, CO, temperature, and pressure are updated. PM2.5 is reconstructed from predicted AQI using EPA breakpoints. Change-rate features are recalculated at each step.

This approach mirrors training conditions, eliminating a 72% error from the previous static-lag method.

# Automated CI/CD Pipeline

Two GitHub Actions workflows fully automate operations.

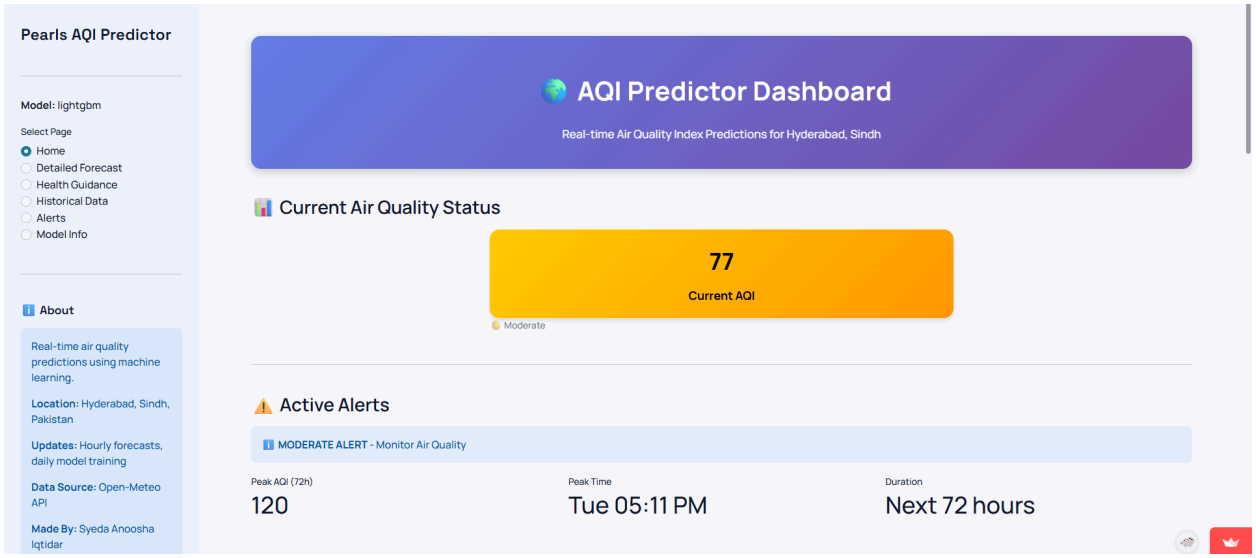
- **Hourly Feature Pipeline:** Runs at 42 minutes past each hour. Sets up the environment, installs dependencies, injects Hopsworks credentials, and executes the feature pipeline. The 42-minute offset avoids top-of-hour congestion.
- **Daily Training Pipeline:** Runs at 8:47 AM UTC. Restores model cache, executes full training and registration, and uploads artifacts with 30-day retention. Both workflows allow manual triggers for debugging.

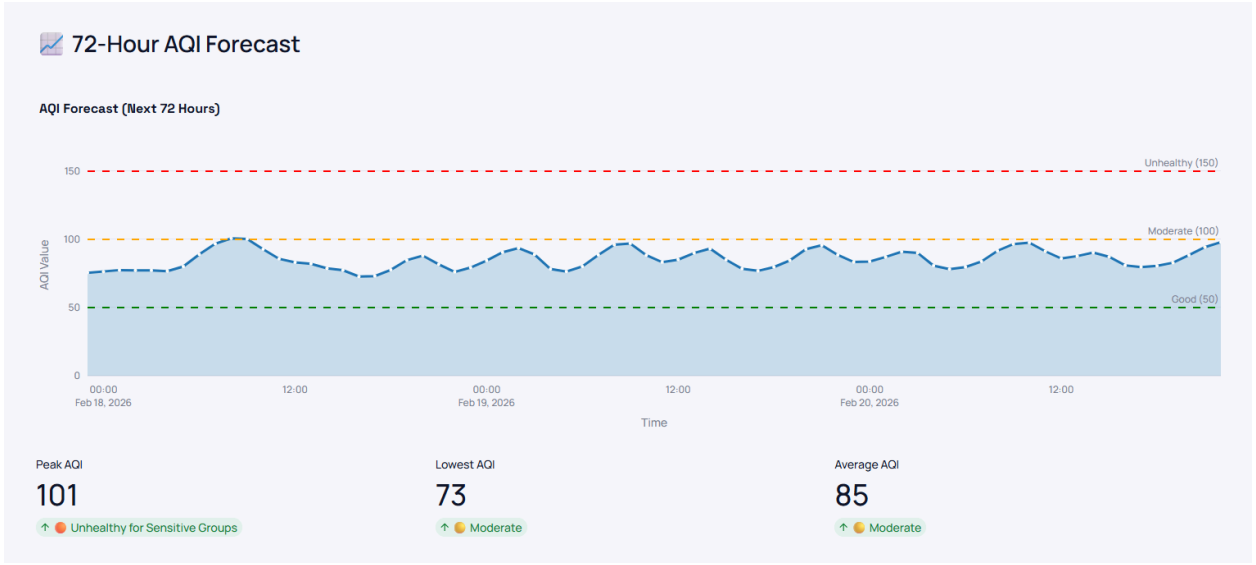
Secrets are managed via GitHub encrypted secrets. Retry logic (3 attempts, linear backoff) handles transient API failures. The system operates within GitHub Actions' free tier limits.

# Web Application Dashboard

The dashboard consists of a FastAPI backend and Streamlit frontend, deployed independently.

- **Backend:** Provides endpoints for current AQI, 72-hour forecasts, health alerts, historical data, SHAP explanations, feature importance, health guidance, and model info. Models load from Hopsworks initially and are cached; later predictions are under 100 ms.
- **Frontend:** Six pages are created, namely Home (current AQI + 72h trend), Detailed Forecast, Health Guidance, Historical Data, Alerts, and Model Info.




# Data Insights and Explainability

## SHAP-Based Prediction Explanation

SHAP decomposes each prediction into feature contributions. For example, a 105 AQI prediction might reflect:

- +12.3 from PM2.5 lag 24h
- +5.7 from the afternoon hour
- +3.2 from high temperature
- -4.1 from moderate win

The global SHAP summary confirms the feature importance hierarchy: lag features (especially PM2.5 at 24h and 1h) dominate, temporal features (hour, month) drive cycles, and weather features (temperature, wind, pressure) have moderate effects.

## Health Alert System

Predicted AQI values are mapped to six EPA categories with color codes, messages, and actionable guidance:

- **Good (0-50, green):** No precautions
- **Moderate (51-100, yellow):** Sensitive individuals limit exertion
- **Unhealthy for Sensitive Groups (101-150, orange):** Children, elderly, asthmatics, reduce outdoor activity
- **Unhealthy (151-200, red):** General population advised N95 masks
- **Very Unhealthy (201-300, purple):** Avoid all outdoor activity
- **Hazardous (301+, maroon):** Emergency measures, stay indoors, seek medical help if needed

The system scans the 72-hour forecast and issues the alert corresponding to the highest predicted AQI category.

# Challenges and Solutions

Building a production AQI predictor revealed a variety of issues across data, modeling, and infrastructure. The key challenges and their resolutions are summarized below.

## Data Leakage from Same-Hour Pollutants

- **Problem**: Using current-hour PM2.5, CO, $NO_2$, and $SO_2$ as features caused a near-perfect $R^2$ (>0.999) because the model had direct access to the target.
- **Solution**: Remove same-hour pollutant columns; retain only lagged values. Unrealistic performance almost always indicates leakage. Features must be scrutinized for any target information.

## Target Leakage through AQI Lag Features

- **Problem**: Including AQI at t-1, t-3, and t-24 produced near-perfect metrics, but failed during abrupt events.
- **Solution**: Drop AQI lags; keep only pollutant and weather vars lags, and introduce change-rate features using strictly past values. Even legitimate-seeming features can encode the target indirectly.

## Rolling Features and Subtle Leakage

- **Problem**: Rolling mean and standard deviation features, like 3-hour and 24-hour averages, are aimed at capturing trends and volatility. However, rolling windows can include the current row's value, leading to overfitting. The model had near-perfect training accuracy but high validation error, indicating overfit.
- **Solution**: All rolling features were removed in favor of pure shift-based lags, which use only data from strictly earlier timestamps. Rolling statistics can introduce subtle, hard-to-detect leakage.

## AQI Change-Rate Target Leakage

- **Problem**: Original formula, aqi[t] - aqi[t-N], embeds the target in features. Feature importance analysis uncovered this: the combined importance of change and rate features was disproportionately large, and short-term PM2.5 lags ranked anomalously low.
- **Solution**: The fix shifted the formula to aqi[t-1] - aqi[t-1-N], using only past values. Training metrics degraded slightly (as expected), but test RMSE actually improved by 2%,  and the validation-to-test gap narrowed from 33% to 25%

## Lag Misalignment and Chronological Sorting

- **Problem**: Creating lag features on unsorted data produced misaligned values: pm2_5_lag_1h at a given row did not correspond to the PM2.5 value from one hour earlier, but rather from the adjacent row, which might represent a completely different timestamp.
- **Solution**: Sort data chronologically and drop initial rows with NaN lags.

## API Failures and Resilience

- **Problem**: Open-Meteo API occasionally timed out. Without retry logic, a single transient failure would leave a gap in the feature store.
- **Solution**: Retry wrapper with linear backoff (3, 6, 9 s). After three retires failures are logged and the pipeline continues with available data.

## Train-Serve Skew in Forecast Generation

- **Problem**: Despite achieving $R^2 = 0.96$ on test data, the live forecast initially showed 20% error because the inference code copied lag values from the last historical row into all 72 forecast hours. By hour 2, the lag features, which make up about 60% of model importance, were already stale.
- **Solution**: Autoregressive loop updating lag buffers hourly; error reduced to 5.5%.

## Lessons and Reflections

Production ML focuses on data engineering, testing, and deployment, not modeling. I found four leakage types: same-hour pollutants, AQI lags, rolling windows, and change-rate targets, showing signs like unrealistic metrics or inverted feature importance. Detecting and fixing leaks is an ongoing part of feature engineering and evaluation, not a one-time task check.

Time-series modeling requires discipline: sort data chronologically, use only past data for lag features, and avoid shuffling targets. Small misalignments cause silent errors. Train-serve skew is another challenge; models with evolving lag features may fail in production if served with static lags.

External APIs are unreliable, so robust retry logic, logging, and caching strategies are necessary for performance and reliability.

Model complexity should match the dataset. Overly deep or unregularized models overfit, while autoregressive inference aligns training and predictions. CI/CD, automation for pipelines, feature validation, and monitoring are essential as small errors compound quickly.