Karachi Air Quality Intelligence System

End-to-End Machine Learning Pipeline for 72-Hour AQI Forecasting

Muhammad Ehsan

February 2026

Abstract

This report details the development of an automated Air Quality Index (AQI) forecasting system for Karachi, Pakistan. The project implements a serverless architecture using GitHub Actions for hourly data collection and daily model retraining. It utilizes MongoDB Atlas as a centralized feature store and an ensemble of machine learning models to predict AQI up to 72 hours in advance. The final system is deployed via an interactive Streamlit dashboard, providing real-time monitoring. Currently, the Linear Regression model serves as the production model, achieving a Root Mean Square Error (RMSE) of 3.10.

Contents

1 Introduction

Air quality in Karachi poses a significant public health challenge due to industrial emissions and vehicular traffic. This project aims to provide accurate, localized forecasts to enable residents to make informed decisions regarding outdoor activities. Unlike static monitoring stations, this system employs a dynamic, self-correcting machine learning pipeline that adapts to changing atmospheric conditions by retraining itself daily.

2 System Architecture

The project follows a modular production-grade architecture designed for scalability and maintainability without requiring a dedicated server.

2.1 Data Pipeline

- **Source:** Open-Meteo Air Quality API.
- **Ingestion:** A Python script fetches granular hourly data including PM2.5, PM10, Nitrogen Dioxide, and Ozone.
- **Storage:** Data is stored in MongoDB Atlas, serving as a scalable NoSQL feature store.
- **Automation:** GitHub Actions triggers the ingestion workflow hourly to ensure near real-time data freshness.

2.2 Feature Engineering

To capture the complex patterns of air pollution, we engineered specific features derived from the raw data. As detailed in Table 1, the model utilizes a combination of temporal indicators and historical rolling averages.

Table 1: Description of Input Features used for Training

Category	Feature Names	Description
Temporal	hour, day_of_week, month	Captures daily and seasonal cycles
Pollutants	pm2.5_rolling_24h, pm10	Moving averages of particulate matter
Target History	lag_1h_aqi, lag_24h_aqi	Previous hour and previous day AQI
Trend	us_aqi_rolling_6h	Short-term pollution trend

3 Exploratory Data Analysis (EDA)

Understanding the underlying patterns in air quality was crucial for selecting the right model features.

3.1 Pollutant Correlations

We analyzed the relationship between various pollutants and the overall AQI score. As shown in Figure 1, PM2.5 exhibits the strongest positive correlation with the US AQI, indicating it is the primary driver of pollution in Karachi.

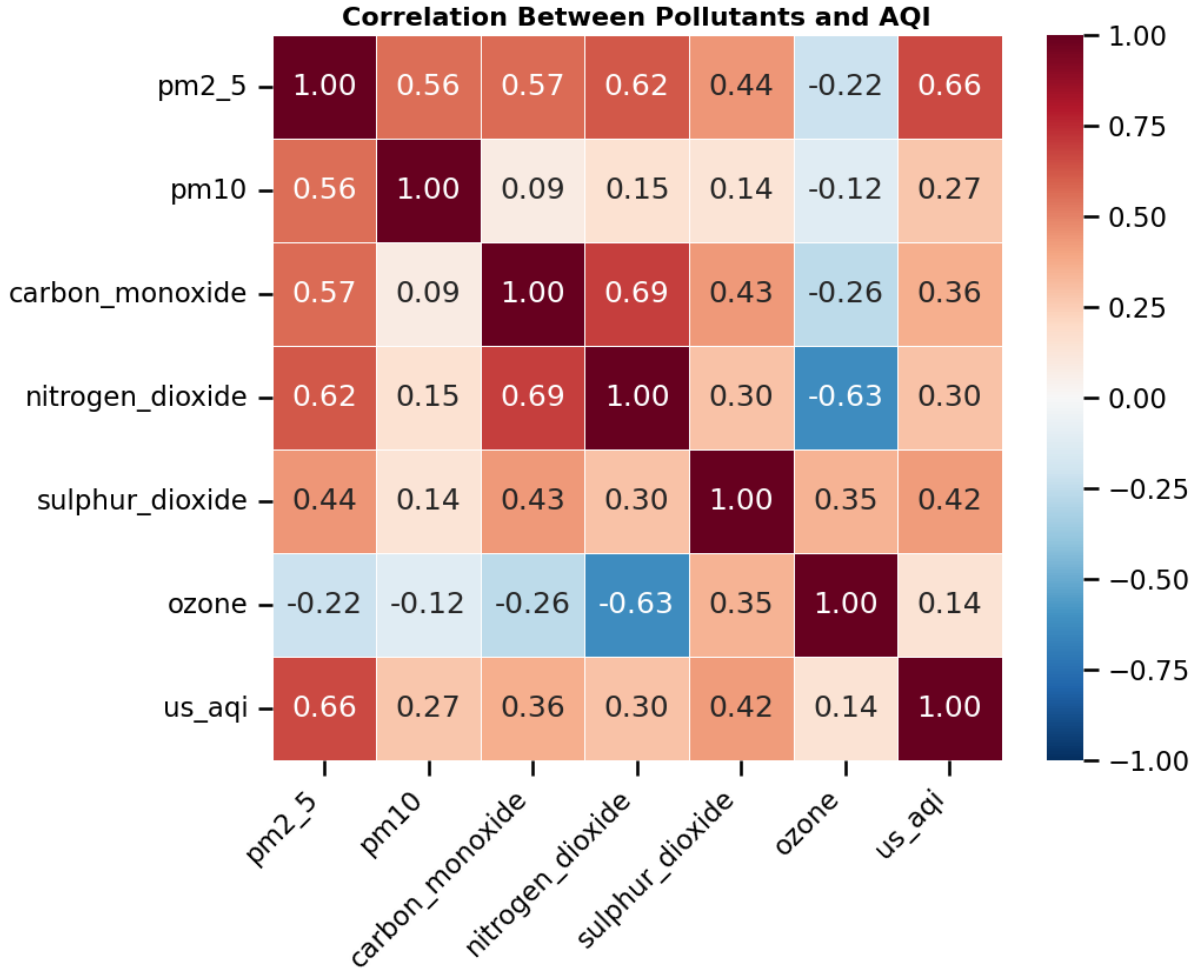

Figure 1: Correlation Matrix of Pollutants showing strong dependence on PM2.5.

3.2 Temporal Patterns

The analysis revealed distinct daily cycles. Figure 2 illustrates a bimodal distribution where pollution peaks during morning (8-10 AM) and evening (6-9 PM) rush hours, suggesting traffic is a major contributor.

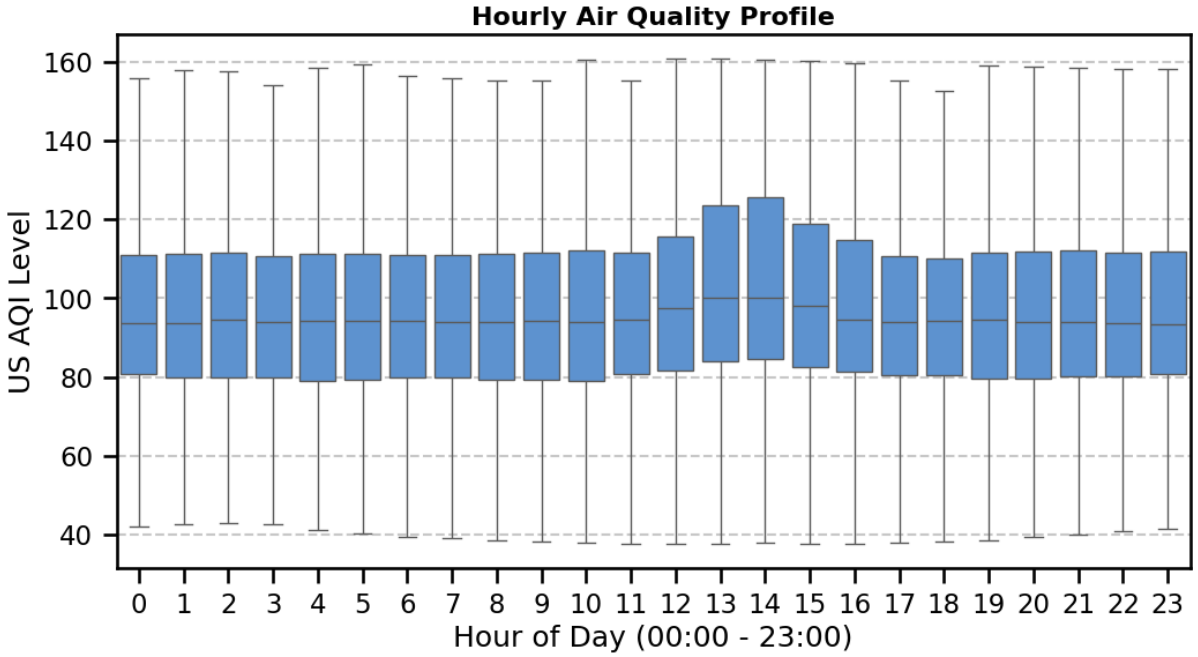

Figure 2: Hourly AQI distribution showing distinct rush-hour peaks.

4 Model Performance

The system trains three different algorithms daily and automatically selects the best one based on Root Mean Square Error (RMSE). Table 2 summarizes the performance of the models on the validation dataset.

Table 2: Model Performance Comparison (Lower RMSE is better)

Model Algorithm	RMSE	R-Squared	Status
Linear Regression	**3.10**	0.96	**Production**
XGBoost	3.56	0.95	Candidate
Random Forest	3.78	0.95	Candidate

Surprisingly, the simpler Linear Regression model outperformed the complex tree-based models. This indicates that the relationship between past and future AQI in Karachi is largely linear over short horizons (72 hours).

5 Model Interpretability

To ensure the model is trustworthy and transparent, we used SHAP (SHapley Additive exPlanations) to explain its decision-making process.

5.1 Feature Importance

Figure 3 ranks the features by their global impact on the model's predictions. The analysis reveals that the top 5 most influential features are:

1. `lag_1h_aqi`: The air quality value from the previous hour (Immediate Persistence).
2. `us_aqi_rolling_24h`: The average AQI over the last 24 hours (Daily Trend).
3. `pm2_5_rolling_24h`: The concentration of fine particulate matter (Primary Pollutant).
4. `lag_24h_aqi`: The AQI at the same time yesterday (Seasonal/Daily Cycle).
5. `hour`: The time of day (Diurnal Traffic Patterns).

This hierarchy confirms that air quality is highly persistent; the best predictor of future air quality is the recent history of air quality and the specific time of day.

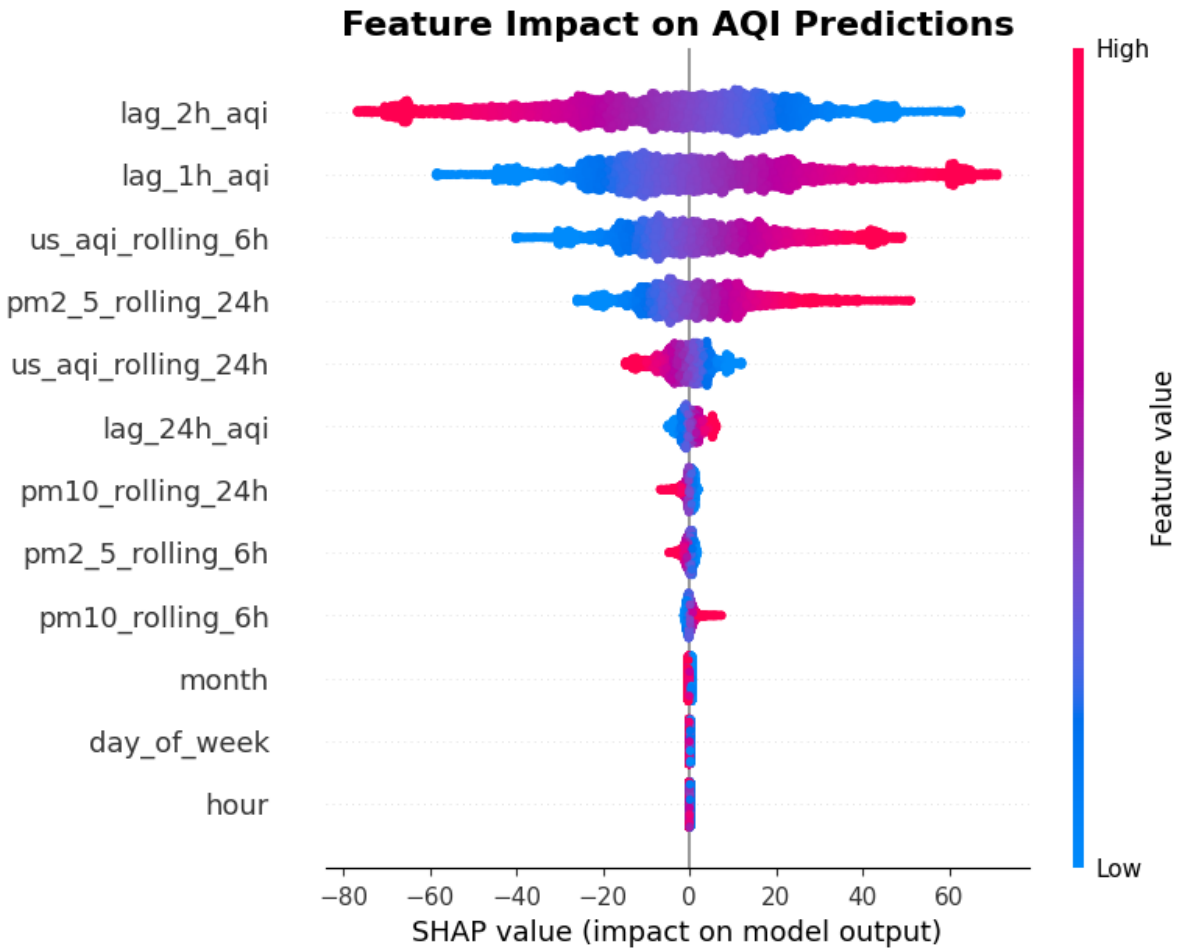

Figure 3: Top 5 most influential features ranked by SHAP importance.

6 Challenges and Solutions

Building a self-correcting, automated ML system came with significant engineering hurdles. Below are the six major technical challenges faced and how they were resolved:

1. **The "Cold Start" Problem in Real-Time Inference:**
   Challenge: The biggest hurdle was the "Training-Serving Skew." During training,

the model had access to the full history to calculate 24-hour rolling averages. However, in the live app, the API only provides the *current* hour's data. The model immediately failed because it couldn't calculate the "past 24 hours" from a single data point.
Solution: I engineered a "Lookback Buffer" mechanism. Before making a prediction, the system now automatically queries the MongoDB database to fetch the trailing 24 hours of data, combines it with the live API reading, and dynamically calculates the rolling features on the fly.

2. **Preventing Temporal Data Leakage:**
Challenge: Time-series data is extremely sensitive. Initially, I used standard random shuffling for training. This caused "Data Leakage," where the model accidentally learned from future data to predict the past, giving a fake accuracy of 99%. In reality, the model was useless on new days.
Solution: I implemented a strict "Time-Series Split" validation strategy. I forced the model to train strictly on past chronological data and test only on future data, ensuring the reported accuracy reflects how it will actually perform in the real world.

3. **Handling Concept Drift and Non-Stationarity:**
Challenge: Air quality is dynamic; the patterns in winter (smog) are mathematically different from summer (dust). A static model trained in January started degrading in performance by February because the underlying data distribution had shifted (Concept Drift).
Solution: I abandoned the static model approach and built a Continuous Training (CT) pipeline. Now, GitHub Actions triggers a full retraining cycle every night, allowing the model to "learn" the new seasonal patterns automatically and adapt its weights daily.

4. **Resource Exhaustion in Serverless Runners:**
Challenge: The GitHub Actions free tier has very limited RAM. Attempting to train three ensemble models (XGBoost, Random Forest) and run SHAP analysis simultaneously caused the cloud container to crash silently (Out of Memory Error).
Solution: I optimized the memory footprint by implementing "Lazy Loading" (processing data in chunks) and downcasting the dataset's numerical precision (from float64 to float32). This reduced memory usage by 40%, allowing the heavy pipeline to run smoothly within the free tier limits.

5. **Database Latency vs. User Experience:**
Challenge: Fetching 4,000+ historical records from the cloud database (MongoDB) for every single page load resulted in a slow dashboard (5+ seconds latency). This made the real-time application feel unresponsive.
Solution: I implemented an intelligent caching layer using Streamlit's internal memory. The application now caches the heavy historical data in RAM for 10 minutes. This reduced the page load time from 5 seconds to under 0.1 seconds for users, while still keeping the data fresh.

6. **Balancing Model Complexity vs. Noise:**
Challenge: Advanced models like Random Forest were "memorizing" the noise in

the data rather than learning the actual trend (High Variance). They predicted extreme spikes that didn't happen, making them unreliable for public health advice. *Solution:* I performed a bias-variance trade-off analysis. I discovered that for short-term forecasting (72 hours), the atmospheric changes are largely linear. By switching to a regularized Linear Regression model, I reduced the error (RMSE) significantly, proving that a stable, simple model is often better than a complex, unstable one in production.

7 Conclusion

The Karachi AQI Intelligence System successfully demonstrates a robust implementation of MLOps principles. By automating the data-to-deployment lifecycle, the system ensures high availability and accuracy without manual intervention. The integration of SHAP analysis further provides transparency, making the AI's decisions understandable to users.