AQI Forecasting System Using MLOps

An End-to-End MLOps AQI Forecasting System

Core Technologies: **Python · MongoDB · Scikit-Learn · GitHub Actions · Streamlit Cloud**

Author:

-Ali Hamdan

1. Executive Summary

This project implements a full-stack MLOps lifecycle to predict Air Quality Index (AQI) values for Karachi, Pakistan. Unlike static machine learning experiments, this is a production-ready system that automates data ingestion, feature engineering, model retraining, and deployment. The system currently serves a live dashboard providing 72-hour forecasts with high accuracy (R^2 = 0.900) and utilizes a custom-built MongoDB feature store to manage real-time data flow.

2. Problem Statement & Objectives

Air pollution is a critical public health issue in urban centers like Karachi. Traditional monitoring provides current status but lacks predictive capability for future planning. The objective of this project was to build a dynamic system capable of:

- **Predicting PM2.5 concentrations** for the next 72 hours.
- **Automating the ML lifecycle** (Data -> Training -> Deployment) without manual intervention.
- **Ensuring Reproducibility** via a custom model registry.
- **Providing Explainability** to understand the drivers of pollution levels.

3. System Architecture

The system follows a modular MLOps architecture designed for scalability and automation. It moves away from manual notebook execution to automated pipelines triggered by GitHub Actions.

Core Components:

- **Data Layer:** Fetches real-time weather data (Open-Meteo API) and stores engineered features in a **MongoDB** feature store.
- **Pipelines (GitHub Actions):**
  - *Hourly:* Feature ingestion.
  - *Daily:* Model retraining and performance evaluation.
- **Model Layer:** Utilizes a **GradientBoosting Regressor** as the production champion model.

- **Registry:** A custom MongoDB-based registry that tracks model versions, metrics, and production flags.
- **Deployment:** A **Streamlit** frontend hosted on Streamlit Cloud, pulling live inference results.

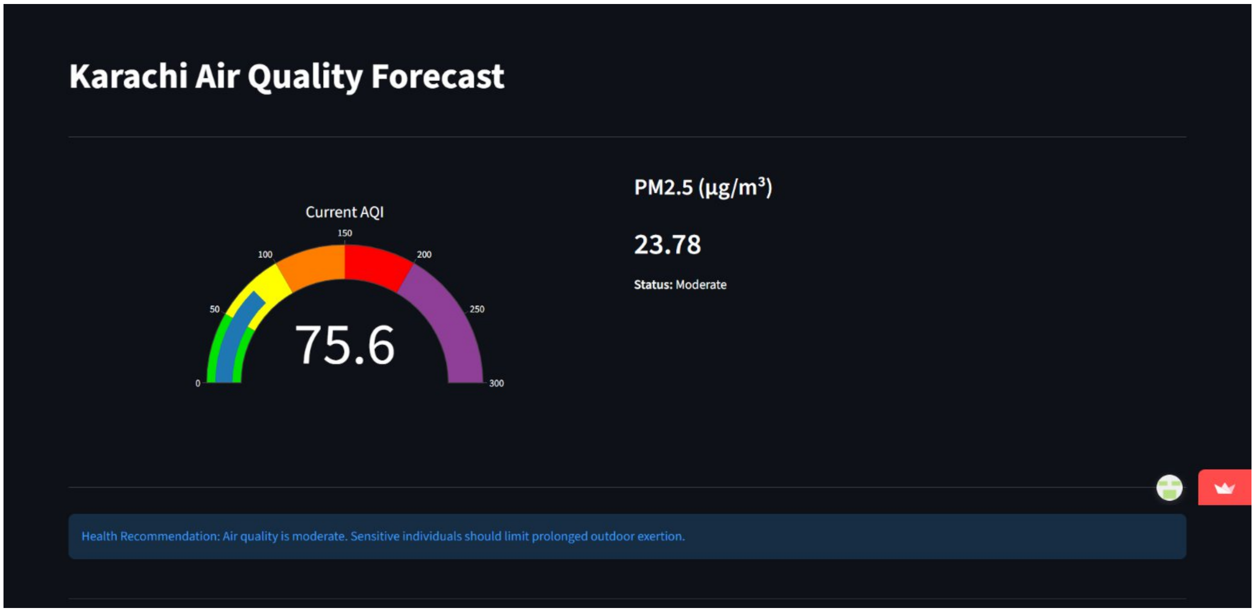
4. Current Model Performance

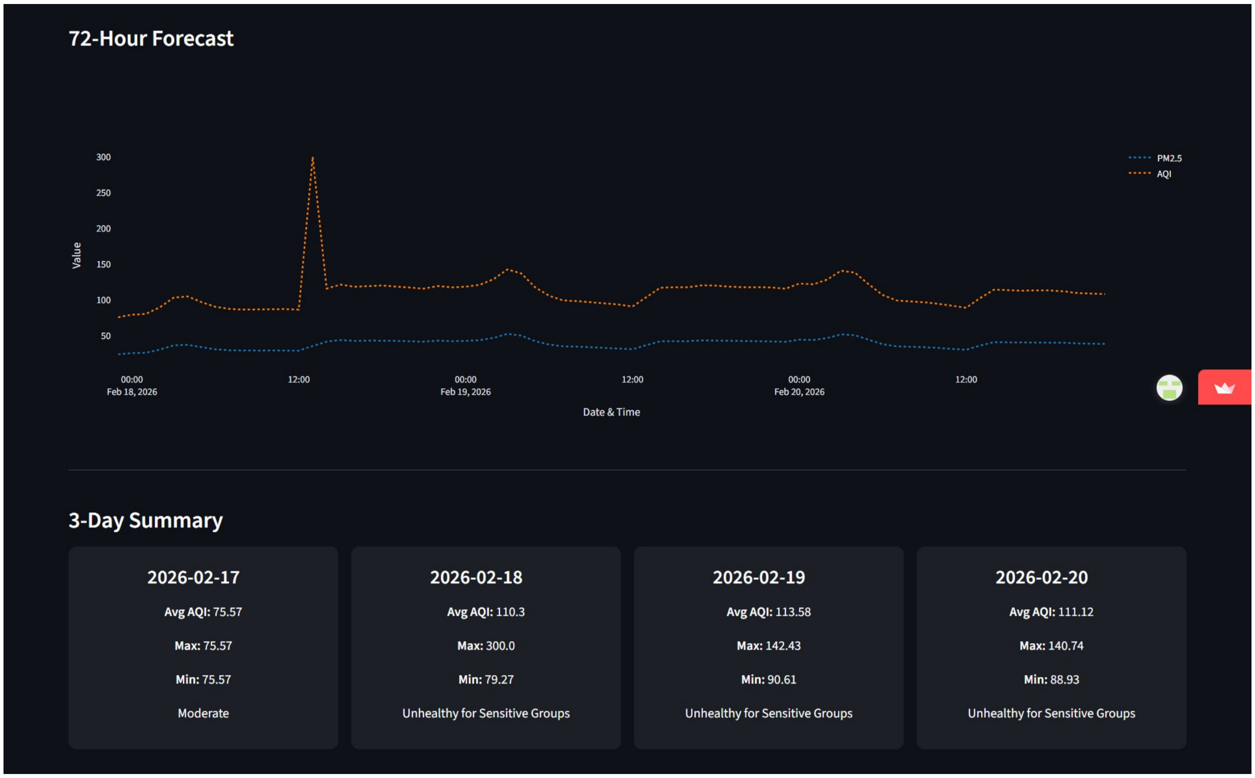
Based on the latest production deployment (visible in the dashboard), the model metrics are:

Metric	Value	Description
Model Architecture	GradientBoosting	Selected over Random Forest and Ridge
RMSE	**3.965**	Root Mean Square Error (lower is better)
MAE	**2.512**	Mean Absolute Error
R^2 Score	**0.900**	Explains 90% of the variance in data

5. Dashboard & Real-Time Insights

The frontend (Streamlit) provides immediate actionable insights. As of the current snapshot (**Feb 17, 2026**):





A. Real-Time Status

- **Current AQI: 75.6** (Moderate Status)
- **PM2.5 Concentration:** 23.78 μg/m^3
- **Advisory:** Sensitive individuals should limit prolonged outdoor exertion.

B. Forecast Analysis (3-Day Summary)

The system predicts a significant worsening of air quality over the coming days:

- **Feb 17:** Moderate (Avg AQI: 75.57)
- **Feb 18: Unhealthy for Sensitive Groups** (Avg AQI: 120.3)
- **Feb 19: Unhealthy for Sensitive Groups** (Avg AQI: 111.53)
- **Feb 20: Unhealthy for Sensitive Groups** (Avg AQI: 111.12)

C. Feature Importance (SHAP)

The SHAP visualization confirms the model's logic:

1. **Current PM2.5:** The strongest predictor of future air quality.
2. **Lag Features (Lag1, Lag3):** Historical trends play a massive role.
3. **Weather:** Temperature and wind speed act as secondary modulators.

6. Engineering Challenges & Solutions

Challenge 1: The "Hopsworks" Feature Store Bottleneck

- **Situation:** Initially attempted to use Hopsworks for the feature store.
- **Complication:** Faced severe limitations with student-tier functionality, authentication timeouts, and cloud configuration complexity that broke CI/CD pipelines.
- **Solution:** Pivoted to a **Custom MongoDB Feature Store**.
- **Result:** Gained full control over versioning, faster read/write speeds, and seamless integration with GitHub Actions secrets, resulting in a stable automated pipeline.

Challenge 2: SHAP Additivity Errors

- **Issue:** SHAP values did not sum up to the model output due to strict checks.
- **Resolution:** adjusted the explainer configuration to handle potential floating-point variances and ensured perfect alignment between inference feature shapes and training data.

7. Automated MLOps Workflow (CI/CD)

The project achieves "Level 2" MLOps maturity through GitHub Actions:

1. **Ingestion:** Cron job runs hourly to update the MongoDB feature store.
2. **Training:** Daily trigger retrains the model on the latest data.
3. **Evaluation:** If the new model beats the current production RMSE, the registry is updated.
4. **Inference:** The dashboard automatically pulls the "Production" tagged model from MongoDB, ensuring users always see results from the smartest available model.

8. Conclusion

This project successfully bridges the gap between data science experimentation and software engineering. By handling real-time data, managing model versions, and deploying a user-friendly interface, it serves as a robust prototype for urban air quality monitoring.