End-to-End AQI Prediction System with Production-Grade MLOps

Muhammad Ansab Chaudhary

1 Introduction

This project presents a production-style, end-to-end Air Quality Index (AQI) prediction system built using a modern MLOps architecture. The system integrates real-time data ingestion, feature engineering, automated training, model registry management, deployment through a Streamlit interface, and model interpretability via SHAP.

The primary objective was not only to build accurate predictive models, but to design a stable, automated, and reproducible ML pipeline suitable for real-world deployment.

2 System Architecture

The system consists of the following components:

- **Data Sources:** OpenWeather Current API, Air Pollution API, and historical weather archives.
- **Feature Store:** Hopsworks Feature Store (Cloud Edition).
- **Model Training:** Automated daily retraining pipeline.
- **Model Registry:** Hopsworks Model Registry.
- **CI/CD:** GitHub Actions for feature and training pipelines.
- **Deployment:** Streamlit dashboard.
- **Interpretability:** SHAP-based feature attribution.

3 Feature Pipeline

The hourly feature pipeline performs:

1. Fetching current weather and pollution metrics.
2. Constructing temporal features (hour, day, month).
3. Computing derived feature: AQI change rate.
4. Inserting rows into Hopsworks Feature Store.

The feature group uses composite primary keys: `city, event_time`. Concurrency control and retry logic were implemented to prevent duplicate inserts and API instability.

4 Exploratory Data Analysis

The exploratory analysis was conducted on approximately 180 days of historical data collected at hourly intervals. Since each day contributes 24 observations, the dataset contains roughly 4,300–4,500 rows in total. This provides sufficient temporal coverage to capture short-term pollutant fluctuations, seasonal variation, and trend behavior in AQI levels.

The dataset includes both meteorological variables (temperature, humidity, pressure, wind speed) and pollutant concentrations (PM2.5, PM10, NO2, O3), along with engineered temporal features (hour, day, month) and a derived AQI change rate feature.

4.1 AQI Distribution

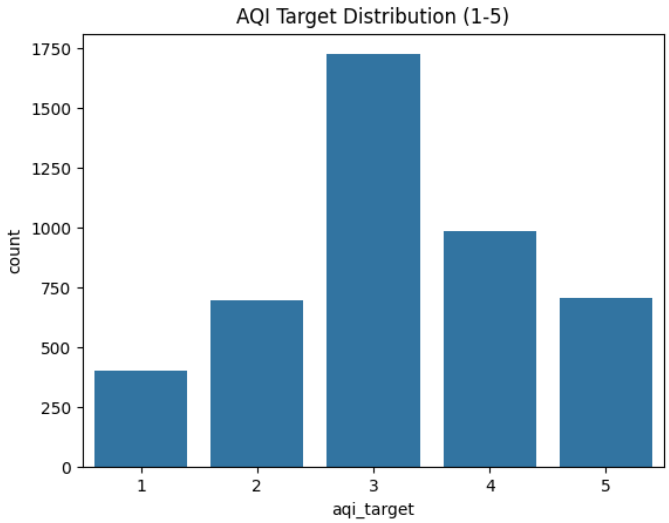

Figure 1: AQI Target Distribution (1-5)

The dataset shows that AQI category 3 (Moderate) dominates the distribution. Categories 4 and 5 are also significant, indicating frequent unhealthy air conditions.

The class imbalance required careful evaluation to avoid biased predictions.

4.2 Feature Correlation

Key observations:

- PM2.5 and PM10 show strong positive correlation with AQI.
- Ozone (O3) also significantly impacts AQI.
- Humidity shows negative correlation.
- Temporal features have weak correlation.

This confirms pollutant concentration as the primary driver of AQI changes.

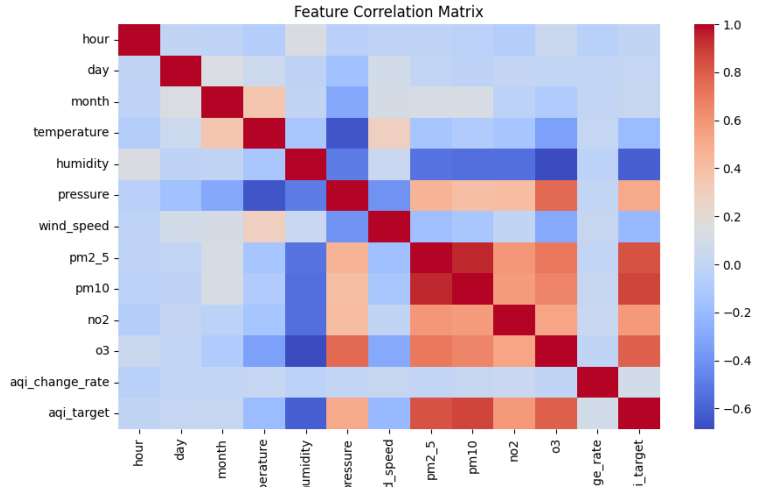

Figure 2: Feature Correlation Matrix

5 Training Pipeline

The daily training pipeline:

1. Reads full historical feature store data.
2. Trains multiple models:
   - Ridge Regression
   - Random Forest
   - Gradient Boosting
   - SVR
3. Evaluates using RMSE, MAE, and R^2.
4. Selects the best model (lowest RMSE).
5. Registers model artifacts in Hopsworks Model Registry.

This ensured reproducibility and automatic model versioning.

6 Model Performance Comparison

Four models were trained and evaluated using RMSE, MAE, and R^2 on the historical dataset.

6.1 Analysis of Results

Key observations:

- Tree-based ensemble models (Random Forest and Gradient Boosting) significantly outperform linear models.

Table 1: Model Performance Comparison

Model	RMSE	MAE	R^2
Ridge Regression	0.488	0.388	0.706
Random Forest	0.132	0.025	0.978
Gradient Boosting	**0.131**	0.064	**0.979**
SVR	0.429	0.349	0.773

- Ridge regression struggles due to nonlinear pollutant interactions.
- SVR improves over Ridge but does not match ensemble performance.
- Gradient Boosting achieves the lowest RMSE (0.131) and highest R^2 (0.979), making it the best model.

The strong performance of ensemble methods suggests nonlinear relationships between pollutants and AQI levels. Pollutant interactions (PM2.5, PM10, O3) appear to drive prediction accuracy beyond what linear models can capture.

Based on RMSE minimization, Gradient Boosting was automatically selected and registered in the Model Registry.

7 Model Explainability using SHAP

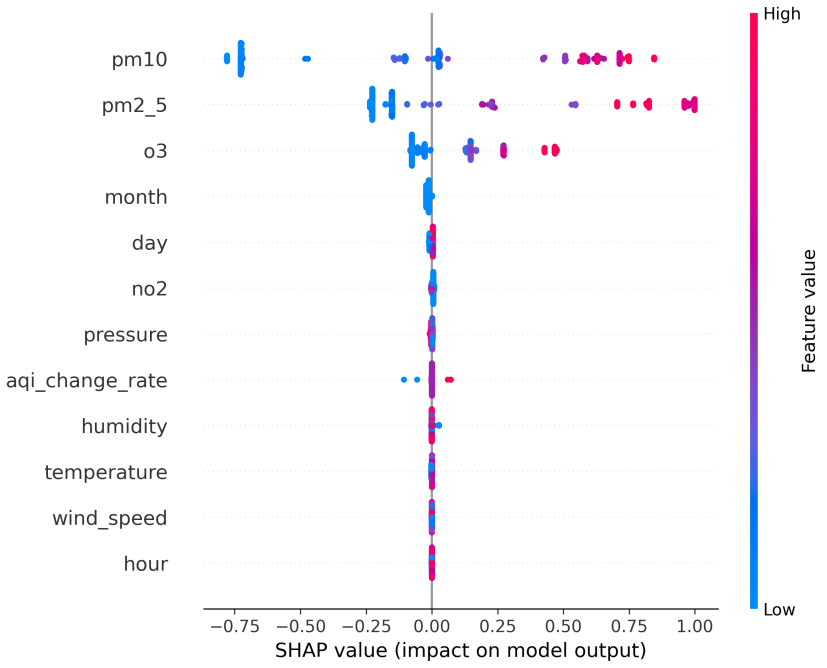

Figure 3: SHAP Summary Plot

SHAP analysis reveals:

- PM10 and PM2.5 are the strongest contributors to prediction output.
- Higher PM values push predictions toward higher AQI classes.
- O3 also contributes positively.
- Weather variables (temperature, wind, pressure) have relatively minor impact.

The SHAP results align with environmental domain knowledge, increasing trust in model predictions.

8 Deployment

The system was deployed using Streamlit Cloud. The dashboard:

- Loads latest model from registry.
- Fetches latest feature row.
- Displays predicted AQI and category.
- Shows 3-day trend (actual vs predicted).
- Visualizes AQI bands.

This completes the full MLOps cycle: data → training → registry → deployment → monitoring.

9 Major Challenges and Failures

9.1 Dependency Conflicts

The most critical issue was dependency incompatibility:

- SHAP upgraded NumPy to version 2.x.
- Hopsworks and PyArrow required NumPy 1.x.
- This caused runtime crashes due to compiled library mismatch.

Resolution required strict version pinning and isolated experimental environments.

9.2 Hopsworks Configuration Issues

Challenges included:

- Authentication token errors.
- Feature group schema mismatches.
- Primary key duplication handling.

Understanding Hopsworks engine initialization and Arrow backend requirements was essential.

9.3 CI/CD Pipeline Complexity

Key difficulties:

- Ensuring reproducibility across local, CI, and Streamlit environments.
- Avoiding breaking deployment while adding experiments.
- Managing multiple requirements files.
- Handling workflow concurrency.

Maintaining strict separation between production and experimental code was critical.

9.4 Model Interpretation Integration

Integrating SHAP required:

- Version alignment with scikit-learn.
- Handling tree-based explainers correctly.

This highlighted the fragility of ML dependency ecosystems.

10 Lessons Learned

- Production ML systems require strict dependency control.
- Feature stores simplify reproducibility but require careful schema management.
- CI/CD integration significantly increases engineering complexity.
- Explainability tools introduce hidden infrastructure challenges.
- Separation of environments prevents production instability.

11 Conclusion

The deployed application is accessible at: `https://aqi-prediction-4180.streamlit.app/`

This project successfully implemented a complete MLOps pipeline for AQI prediction, including automated ingestion, retraining, registry management, deployment, and model interpretability.

The primary learning outcome was not model accuracy alone, but the engineering discipline required to build stable, reproducible, and production-grade ML systems.