Pearls AQI Predictor

Internship Technical Report

Submitted By: Shaheer jamal

Domain: Data Science

Cohort: 7

Executive Summary

This report documents my internship project developing a serverless, automated air quality forecasting system for Karachi, Pakistan. The system predicts Air Quality Index (AQI) up to 72 hours in advance using a hybrid machine learning approach that combines gradient boosting models (LightGBM, CatBoost) and deep learning (LSTM).

The project leverages Hopsworks as a serverless feature store and model registry to ensure reproducible ML workflows. I implemented SHAP-driven feature selection to identify the most critical AQI predictors (24-hour pollutant lags and 6-hour rolling trends), employed time series cross-validation to prevent overfitting, and developed a recursive autoregressive forecasting strategy with seasonal persistence for accurate 72-hour predictions. The system uses RMSE as its primary evaluation metric to disproportionately penalize large forecasting errors, which pose the highest health risks during extreme pollution events.

The deliverable is a deployed Streamlit application providing interactive air quality forecasts, supported by automated backend pipelines for continuous data ingestion, feature engineering, parallel model training, and intelligent model selection—all operating without dedicated infrastructure management.

Chapter 1: Data Acquisition

1.1 The API Journey

The project required all data to come through external APIs. I investigated multiple providers:

Attempt 1: AQICN API

Ground station aggregation for Karachi. Failed due to sparse coverage, 40% missing historical values, and inconsistent timestamps.

Attempt 2: OpenAQ API

Global air quality database with historical data. Provided reliable AQI values but lacked individual pollutant breakdowns (PM2.5, PM10, NO2, etc.) needed for feature engineering.

Final Selection: OpenWeatherMap Air Pollution API

Met all requirements with historical data , granular pollutant sensors (PM2.5, PM10, NO2, SO2, CO, O3, NH3), hourly granularity.

Dataset: 17,000+ hourly observations spanning 2024–2026.

Chapter 2: Exploratory Data Analysis

2.1 Feature Engineering

I systematically transformed raw temporal and pollutant data into a comprehensive feature set designed to capture the complex dynamics of air quality variation. The engineering process focused on extracting temporal patterns, preserving historical context, and encoding cyclical behaviors that influence pollution levels.

Temporal Features:

I extracted foundational time-based attributes from the datetime_utc timestamp, including hour, day_of_week, month, and year. To capture behavioral patterns, I created is_weekend, a binary flag identifying Saturdays and Sundays when traffic patterns and industrial activity differ significantly. Additionally, I encoded season as a categorical variable (winter, spring, summer, fall) to capture quarterly meteorological and emission patterns specific to Karachi's climate.

Lag Features (Capturing Temporal Dependencies):

Time series forecasting relies heavily on autoregressive patterns, the principle that current air quality strongly correlates with recent past values. I engineered multiple lag features at strategic intervals: aqi_lag_1hr, aqi_lag_3hr, aqi_lag_6hr, and aqi_lag_24hr to capture short-term persistence and daily cyclical effects. Similarly, I created PM2.5-specific lags (pm2_5_lag_1hr, pm2_5_lag_24hr) since fine particulate matter is the dominant contributor to AQI in urban environments and exhibits distinct temporal decay patterns.

Rolling Statistics (Capturing Trends):

To identify emerging trends and smooth out short-term noise, I computed rolling window aggregations. Features like aqi_rolling_6h and aqi_rolling_24h represent moving averages that help the model distinguish between temporary spikes (e.g., isolated traffic congestion) and sustained deterioration (e.g., industrial emissions or weather stagnation). The same windowing approach was applied to PM2.5 concentrations (pm2_5_rolling_6h, pm2_5_rolling_24h), providing the model with context about both immediate and sustained pollution trends.

Cyclical Encoding (Handling Seasonality):

Raw hour and month values are inherently cyclical, hour 23 is closer to hour 0 than to hour 12, and December is closer to January than to June. To preserve these relationships mathematically, I applied sine-cosine transformations: hour_sin and hour_cos encode the 24-hour daily cycle, while month_sin and month_cos capture annual seasonal patterns. This encoding prevents models from incorrectly treating cyclical time as linear ordinal data, ensuring that recurring daily and seasonal pollution patterns are properly learned.

Target Transformation:

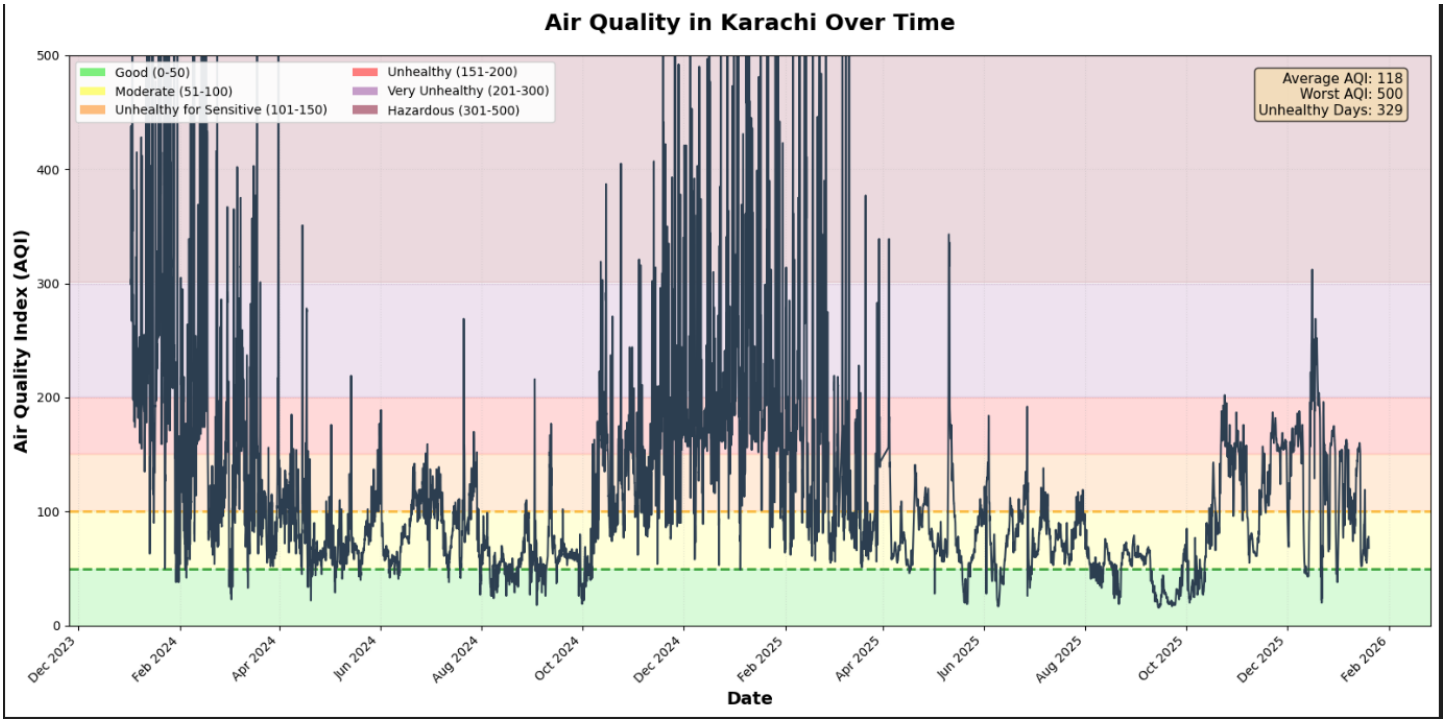
I converted raw pollutant concentrations into standardized AQI values using the US EPA scale (0-500), which maps pollution levels to clinically meaningful health risk categories (Good, Moderate, Unhealthy, etc.). This transformation ensures predictions are directly interpretable for public health decision-making and align with internationally recognized air quality standards.

2.2 Visualizations

I created multiple visualizations to understand Karachi's air quality patterns:

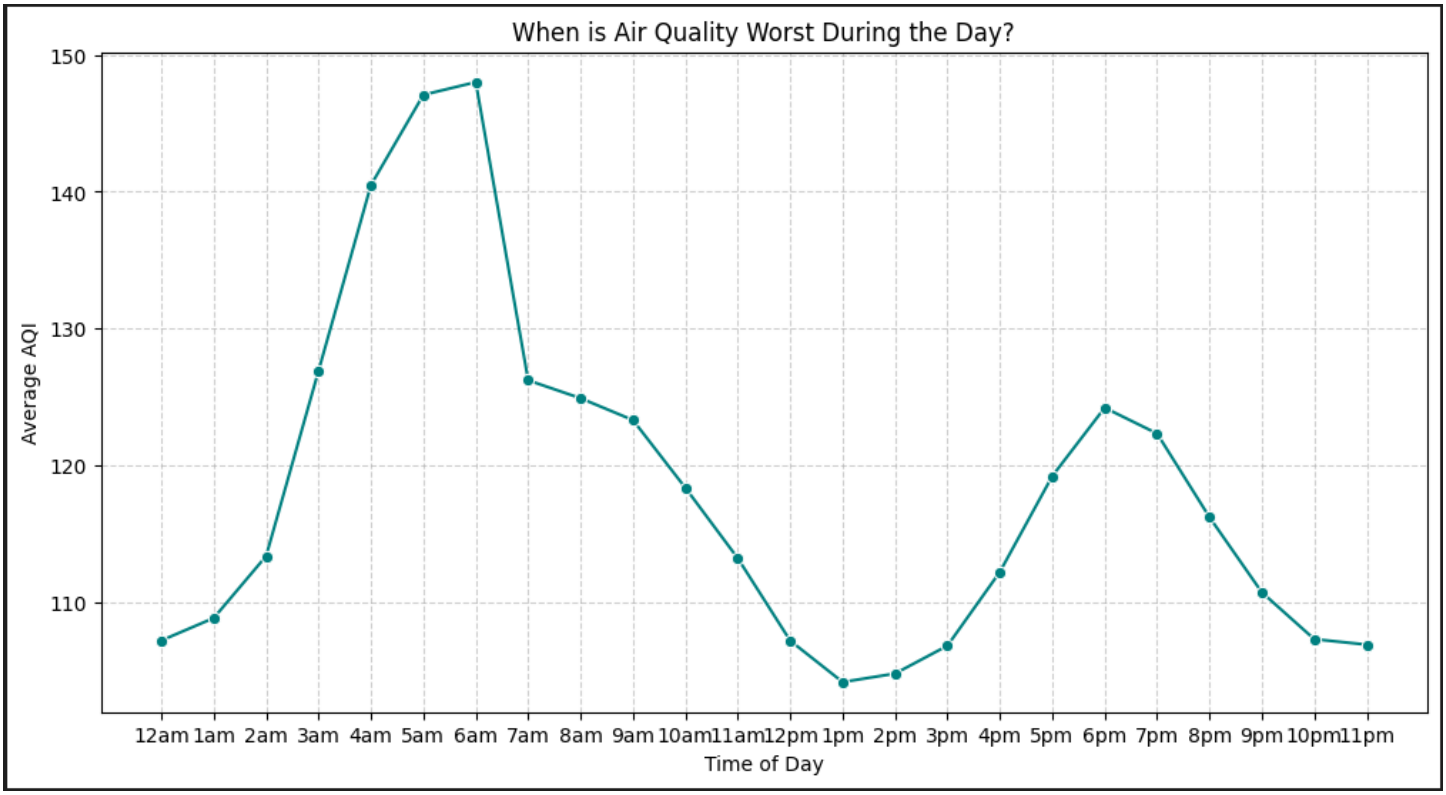
Air Quality Over Time:

Revealed severe winter pollution spikes and cleaner monsoon periods.



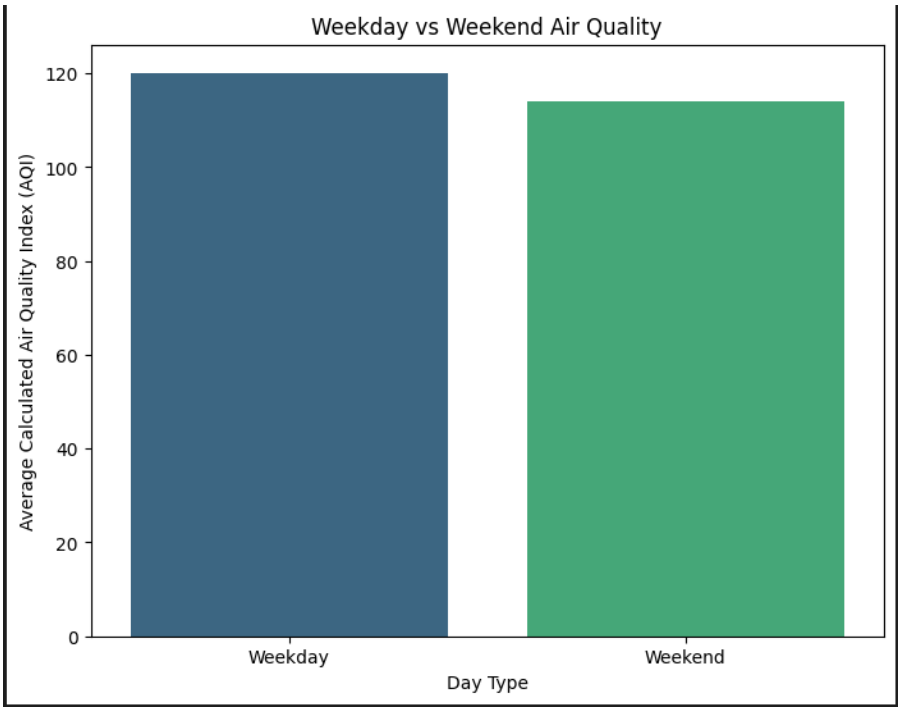
AQI Intensity Over Single Day:

Clear bimodal distribution with morning (8–10 AM) and evening (6–9 PM) peaks correlating with traffic patterns.



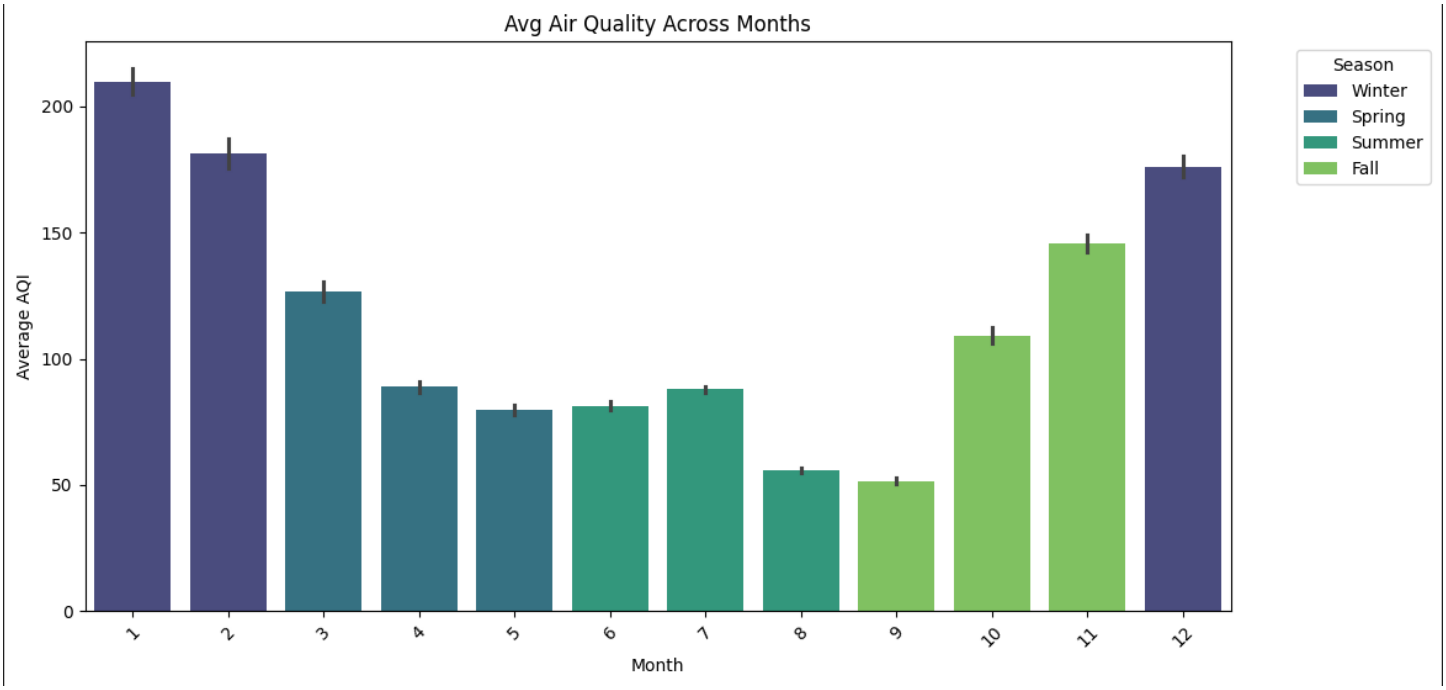
Weekday vs. Weekend Comparison:

Weekends showed slightly lower AQI on average.



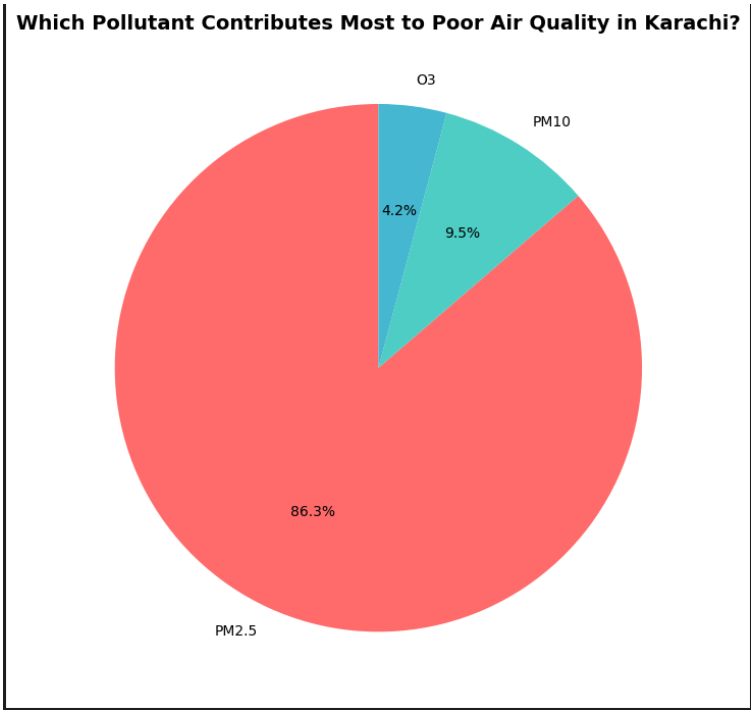
Monthly Averages:

January consistently worst (~210 AQI), September best (~65 AQI).



Pollutant Contribution Pie Chart:

PM2.5 dominated (86% of AQI calculation), followed by PM10 (9%).



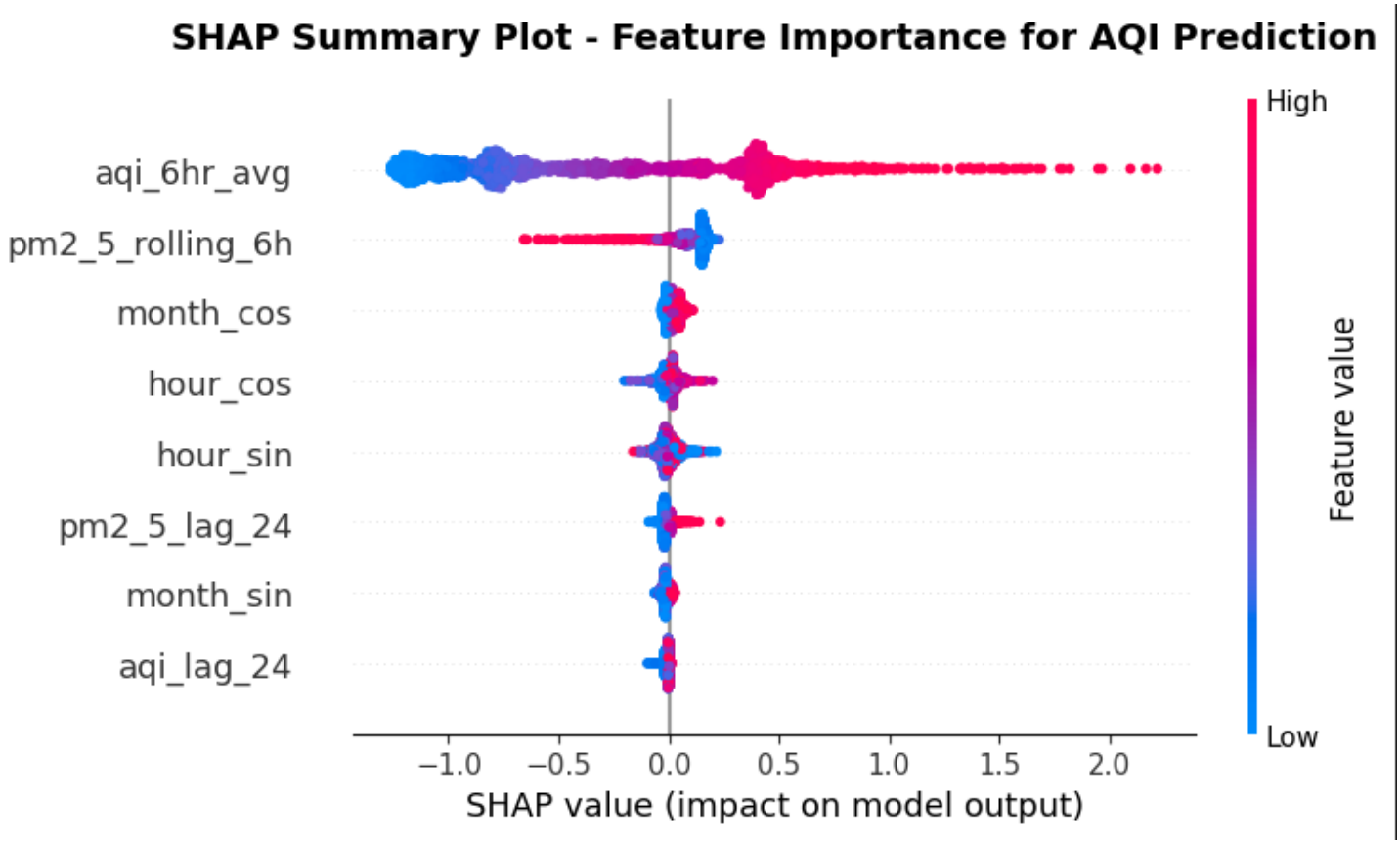
Chapter 3: SHAP Analysis & Feature Selection

3.1 Initial Analysis & The Leakage Discovery

Initial SHAP analysis showed an almost perfect model with R 2 =0.9989 . The SHAP summary plot revealed that pm2_5 (current hour) was absorbing all predictive importance.

The Problem:

I had committed a target leakage. AQI is calculated from pollutant concentrations, including PM2.5. By including current-hour PM2.5, the model wasn't predicting AQI, it was reverse-engineering the AQI calculation formula. For a 72-hour forecast, future pollutant data won't exist, making these features useless in production.



3.2 Refined Feature Selection

I performed aggressive feature pruning:

Removed (Leakage Sources):

All current-hour pollutant readings (PM2.5, PM10, NO2, SO2, CO, O3, NH3)

Short lags (1hr, 3hr) that might proxy current conditions

Retained (Predictive Signals):

Category	Features	Rationale
Atmospheric Momentum	pm2_5_rolling_6h, aqi_6hr_avg	Recent trend without point-in-time leakage
Diurnal Periodicity	pm2_5_lag_24, aqi_lag_24	Captures daily seasonality (traffic, weather)
Temporal Anchors	hour sin/cos, month sin/cos	Encodes cyclical patterns without discontinuity

Dimensionality Reduction:

Low-impact features (nh3, so2, no, is_weekend) were pruned to reduce noise and prevent error accumulation during recursive forecasting.

Chapter 4: Model Development

4.1 Model Selection

LightGBM

Histogram-based gradient boosting with GOSS (Gradient-based One-Side Sampling) and EFB (Exclusive Feature Bundling). Fast training (~30 seconds) fits well within serverless pipeline time limits.

CatBoost

Gradient boosting with ordered boosting to prevent target leakage. Symmetric trees reduce overfitting, and built-in missing value handling accommodates sensor gaps without imputation.

LSTM

Long Short-Term Memory neural network with memory cells and gating mechanisms. Captures non-linear temporal dependencies and complex interactions between meteorological and pollutant variables that tree models miss.

4.2 Training Pipeline

Implemented automated training through GithubActions:

Training Pipeline (runs every 24 hours)

├── Fetch latest data from Feature Store

├── Feature Engineering (lag creation, rolling averages)

├── Train LightGBM, CatBoost, LSTM in parallel

├── Evaluate on TimeSeriesSplit CV

├── Select best model by RMSE and checking overfitting

└── Register to Model Registry

4.3 Model Selection Strategy

The pipeline automatically evaluates all trained models and promotes the best performer.

Primary Metric: RMSE (Root Mean Squared Error)

Rationale: The squaring term in RMSE disproportionately penalizes large errors. In air quality forecasting, missing an extreme pollution spike (e.g., predicting 100 when actual is 300) poses severe health risks. RMSE ensures that the model prioritizes accuracy during hazardous conditions.

Chapter 5: 72-Hour Forecasting Implementation

5.1 Recursive Autoregressive Strategy

Predicting 72 steps ahead requires careful error management. I implemented:

Algorithm:

- **Initial Prediction**: Use actual historical data to predict t+1
- **Recursive Feeding**: Append the t+1 prediction to the input history and use it to predict t+2
- **Iterative Propagation**: Continue this process for 72 iterations, with each new prediction becoming part of the "known" history for subsequent steps
- **Critical Innovation**: Instead of only using the most recent predictions, the algorithm looks back 24 hours within the prediction buffer itself. This means when predicting hour 30, the model references the prediction made for hour 6 (30-24), explicitly preserving diurnal seasonality such as morning traffic peaks and nighttime thermal inversions.

Why Seasonal Persistence Matters: Simple recursive forecasting flattens to the mean over long horizons. By carrying forward 24-hour seasonal patterns, the model maintains realistic daily oscillations instead of converging to a straight line.

Chapter 6: Serverless Architecture & Automation

6.1 Hopsworks Integration

Feature Groups:

The system utilizes a single active Feature Group: aqi_features_karachi (version 2), which stores raw pollutant sensor data (PM2.5, PM10, NO2, SO2, CO, O3, NH3) alongside calculated AQI values. To optimize storage efficiency and maintain flexibility, engineered features, including lag variables, rolling statistics, and cyclical encodings (sin/cos transformations), are not persisted in Hopsworks. Instead, they are computed on-the-fly during both training (scripts/training_pipeline.py) and inference (src/core.py), ensuring the feature engineering logic remains centralized and consistent across the pipeline.

Model Registry:

All trained models are registered in the Hopsworks Model Registry as versioned artifacts under the name aqi_predictor_best, with comprehensive performance metrics including rmse, mae, r2, and train_rmse. The training pipeline evaluates three candidate architectures (LightGBM, CatBoost, LSTM) and automatically selects the winner based on test RMSE while penalizing overfitting. This winner is registered as a new version, effectively implementing implicit promotion to production—the inference service always loads the highest version number, ensuring the most recent best-performing model is deployed without manual intervention.

Overfitting Detection:

The system includes robust overfitting detection logic that calculates an overfitting_gap defined as Test RMSE - Train RMSE. If this gap exceeds a threshold of 10, the model is flagged with an "OVERFITTING WARNING." Critically, the selection algorithm prioritizes models without this warning, even if their raw test RMSE is marginally higher, ensuring deployment stability and generalization capability take precedence over potentially deceptive performance gains from memorization.

6.2 GitHub Actions Automation

Feature Pipeline (Hourly):

```yaml
on:
  schedule:
    - cron: '0 * * * *'
```

Fetches 1 hour of raw data and appends to Feature Store.

Training Pipeline (Every 24 Hours):

```yaml
on:
  schedule:
    - cron: '0 0 * * *'
```

Fetches raw history, engineers features with full context, trains models, and registers best performer.

6.3 Streamlit Frontend

The deployed Streamlit application provides an interactive dashboard for air quality monitoring and forecasting:

- Real-time AQI Status: Displays the current Air Quality Index as a prominently featured metric with health category classification
- 72-Hour Forecast Visualization: Presents future AQI predictions through an interactive bar chart showing the complete 3-day trend
- PM2.5 Concentration Monitoring: Tracks fine particulate matter levels, the primary driver of AQI in urban environments.
- Model Performance Metrics: Features a comprehensive performance card displaying Train RMSE, Test RMSE, MAE, and R^2 Score, along with the active model type (LightGBM, CatBoost, or LSTM). The dashboard includes intelligent overfitting detection that calculates the generalization gap and displays interpretive status messages ("Good Fit" or "Overfitting Risk") to provide transparency about model reliability

Chapter 7: Validation & Quality Assurance

7.1 Overfitting Detection

The pipeline implements robust overfitting detection by calculating the absolute performance gap between training and test sets. The logic is straightforward:

pythongap = test_rmse - train_rmse

if gap > 10:

    flag_overfitting_warning()

This approach directly penalizes models where test performance degrades by more than 10 RMSE units compared to training performance, indicating the model has memorized training patterns rather than learned generalizable relationships. Models flagged with this warning are deprioritized during selection, even if their raw test RMSE appears competitive, ensuring deployment stability over potentially misleading performance metrics.

7.2 Validation Strategy

The system employs a chronological train/test split with an 80/20 ratio, preserving temporal ordering to prevent data leakage. The training set comprises earlier observations (2024–early 2025), while the test set contains recent data (late 2025–2026), simulating real-world deployment where the model must predict genuinely unseen future conditions. Model selection is based on this single held-out test performance combined with the overfitting detection logic, ensuring the chosen model demonstrates both strong predictive accuracy and reliable generalization to recent air quality patterns.

Chapter 8: Challenges & Solutions

This section consolidates all major challenges encountered during the project and their resolutions.

Challenge 1: Data Source Reliability

Problem:

Initial APIs (AQICN, OpenAQ) had insufficient coverage or granularity for production use.

Impact:

Could not build reliable models with 40% missing values or aggregated AQI-only data.

Solution:

Evaluated three APIs systematically, selected OpenWeatherMap for comprehensive sensor data and reliability.

Lesson:

Data quality assessment must precede model development.

Challenge 2: Target Leakage in Feature Engineering

Problem:

Initial model from SHAP analysis achieved $R^2 = 0.9989$ by using current-hour PM2.5, effectively reverse-engineering the AQI formula rather than predicting future values.

Impact:

Model would fail completely in production where future pollutant data is unavailable.

Solution:

- Removed all current-hour pollutant features
- Eliminated short lags (1hr, 3hr) that proxy current conditions

- Retained only lag-24 and rolling features that capture momentum without leakage
- Re-validated with SHAP analysis to confirm feature importance redistribution

Lesson:

Perfect metrics indicate pipeline flaws, not model excellence. Always question "too good to be true" performance.

Challenge 3: Feature Engineering Context in Serverless Pipelines

Problem:

Hourly feature pipeline extracted only 1 row of data, but lag features (e.g., rolling_6h, lag_24) require 6-24 rows for valid calculation.

Impact:

New features were mostly NaN or incorrectly computed, breaking model training.

Solution:

- Refactored architecture to shift feature engineering from hourly ingestion to training pipeline
- Hourly pipeline now appends raw data only
- Training pipeline fetches historical context and performs feature engineering with complete temporal windows

Lesson:

Serverless constraints force architectural trade-offs. Feature engineering timing must respect data dependencies.

Challenge 4: Error Accumulation in Recursive Forecasting

Problem:

72-step recursive prediction without anchoring causes forecasts to converge to mean, losing realistic daily patterns.

Impact:

Long-horizon predictions became useless flat lines.

Solution:

Implemented seasonal persistence by looking back 24 hours in prediction buffer

Maintains diurnal seasonality (morning/evening peaks) across 3-day horizon

Lesson:

Multi-step forecasting requires explicit seasonality preservation mechanisms.

Conclusion & Outcomes

This project successfully delivered a production-grade, serverless air quality forecasting system for Karachi, providing actionable 72-hour AQI predictions through an automated machine learning pipeline deployed as a Streamlit application.

Key Achievements:

The hybrid modeling approach combining gradient boosting (LightGBM, CatBoost) and deep learning (LSTM) ensures robust performance across diverse pollution patterns. SHAP analysis confirmed that 24-hour pollutant lags and 6-hour rolling trends are the dominant predictors, enabling efficient feature engineering. The recursive autoregressive strategy with seasonal persistence maintains realistic daily oscillations throughout the 72-hour forecast, avoiding the common regression-to-mean problem in multi-step predictions.

The serverless Hopsworks architecture automates the complete ML lifecycle, from hourly data ingestion via OpenWeatherMap API to parallel model training, RMSE-based selection with overfitting detection, and seamless deployment. The implicit promotion mechanism ensures the best-performing model is always in production without manual intervention.