📄 Project Report: End-to-End MLOps Pipeline for Air Quality Index (AQI) Forecasting

Author: Muhammad

Date: February 15, 2026

Subject: 10Pearls AQI Project

Project Link: [AQI Forecast (3-Day) · Streamlit]

---

1. Executive Summary

This project focuses on developing an automated, serverless Machine Learning (ML) pipeline to predict the **Air Quality Index (AQI)** for Karachi, Pakistan, up to 3 days in advance. As Karachi consistently ranks among the most polluted cities globally, accurate forecasting is critical for public health.

The system is built on a robust **MLOps architecture** leveraging **Hopsworks Feature Store**, **GitHub Actions**, and **Streamlit**. It automates the entire lifecycle: data ingestion, feature engineering, model training, evaluation, and deployment.

Unlike traditional static models, this system implements a **Dynamic Model Selection Framework**. In every training cycle, five distinct regression algorithms (Random Forest, XGBoost, LightGBM, Linear Regression, and a Voting Ensemble) compete against each other. The pipeline automatically evaluates their performance on a held-out test set and promotes the model with the lowest **Root Mean Square Error (RMSE)** to "Champion" status. This ensures the system always serves the most accurate predictor available, adapting to changing environmental patterns without human intervention.

---

2. Problem Statement

The Context:

Karachi faces severe air pollution challenges, driven by vehicular emissions, industrial activity, and weather patterns. Citizens lack access to localized, data-driven forecasts that predict pollution trends days in advance. Existing weather apps often provide only current readings without predictive insight into how AQI levels will evolve over the next 24-72 hours.

The Objective:

To build a **Self-Correcting MLOps System** that:

1. Ingests real-time weather and pollution data.
2. Trains multiple advanced ML models daily.
3. Dynamically selects the best-performing model based on live metrics.
4. Visualizes future AQI trends via a public dashboard to empower citizens.

---

3. System Architecture

3.1. High-Level Overview

The system is designed as a **Serverless, Batch-Oriented MLOps Pipeline**. Unlike traditional monolithic applications, this project decouples data engineering, model training, and inference into separate, automated workflows. The architecture leverages **Hopsworks** as the central "Feature Store" to ensure consistency between training and inference data, while **GitHub Actions** handles the orchestration of daily tasks.

The architecture consists of five distinct layers:

1. **Data Ingestion Layer** (Extract & Load)
2. **Feature Storage Layer** (Hopsworks Feature Store)
3. **Model Training Layer** (Champion/Challenger Strategy)
4. **Batch Inference Layer** (Forecasting)
5. **Presentation Layer** (Streamlit Dashboard)

3.2. Component Breakdown

Layer 1: Data Ingestion & Transformation

- **Source:** The system consumes data from the **Open-Meteo API**, fetching both Historical Weather Data and 3-day Forecasts.
- **Automation:** A GitHub Action workflow (daily_feature_pipeline.yml) triggers every hour.
- **Processing:**
    - **Cleaning:** Removal of null values and timestamp standardization.
    - **Feature Engineering:** This is where raw data is transformed into model-ready features. Key operations include generating **Lag Features** (e.g., pm2_5_lag_24), **Rolling Averages** (roll_3), and **Cyclical Time Encodings** (hour_sin, hour_cos).
    - **Ingestion:** The processed dataframe is pushed to the Hopsworks Feature Group (aqi_features_main, version 2).

Layer 2: The Feature Store (Hopsworks)

Hopsworks serves as the "Single Source of Truth." It bridges the gap between the data engineering and data science components.

- **Feature Groups:** These act as the physical storage for the data. We maintain a primary group (aqi_features_main) that holds the historical logs of weather and pollution.
- **Feature Views:** These are logical views used to create point-in-time correct training datasets. The system dynamically generates a Feature View (aqi_view_main) to ensure the model doesn't accidentally train on future data (Data Leakage prevention).

Layer 3: Model Training (Champion vs. Challenger)

This layer is responsible for the intelligence of the system.

- **Trigger:** Executed automatically via the **unified daily workflow** (daily_training_prediction.yml). This ensures that the model is retrained every 24 hours on the latest data before generating forecasts.
- **Dynamic Selection:** The training script instantiates five candidate algorithms: **Random Forest**, **XGBoost**, **LightGBM**, **Linear Regression**, and a **Voting Regressor**.
- **Evaluation:** The system splits data into Train (80%) and Test (20%) sets. It calculates **RMSE** for all models.
- **Model Registry:** The winning model (currently the **Voting Regressor**) is serialized (pickled) and saved to the Hopsworks Model Registry. Metadata such as the input schema and performance metrics are attached to the model artifact for lineage tracking.

Layer 4: Batch Inference (Forecasting)

Since Air Quality trends do not change second-by-second, a **Batch Inference** strategy is used.

- **Trigger:** Runs as a **sequential step** immediately following the model training job within the same daily_training_prediction.yml workflow.
- **Process:**
    1. Downloads the "Champion" model from the Registry.
    2. Fetches the "Batch Data" (Recent 24 hours of pollution + Next 3 days of weather forecasts).
    3. Generates predictions for the next 72 hours.
    4. Saves these predictions to a separate Feature Group (aqi_predictions_daily) for the dashboard to consume.

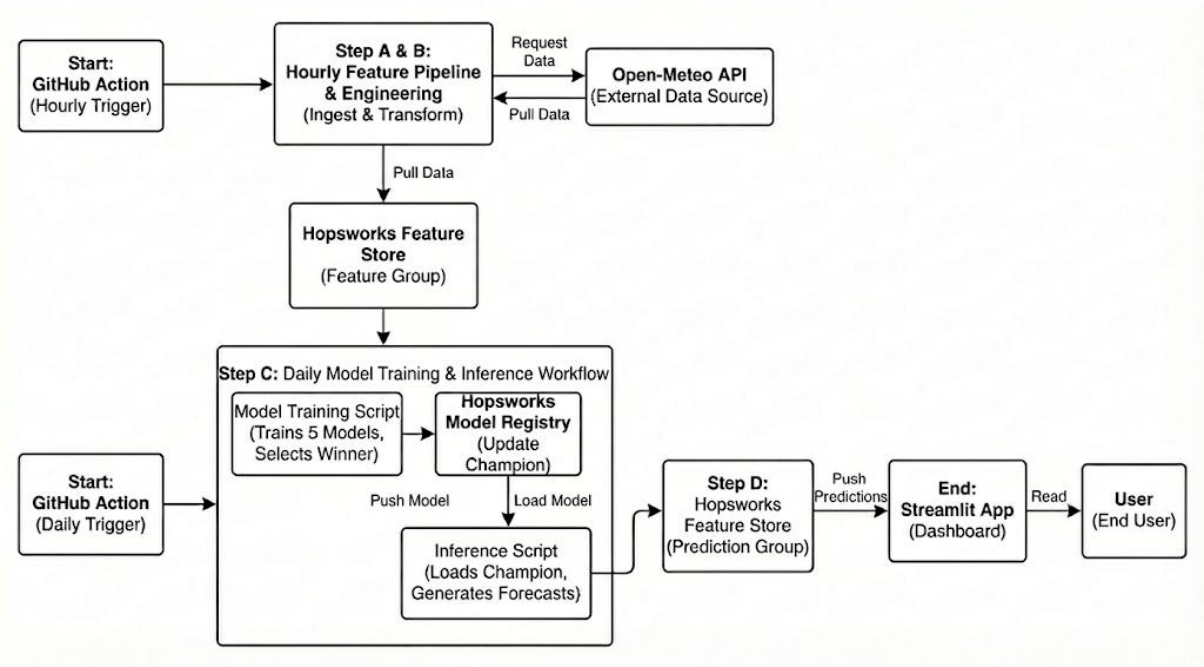
Layer 5: User Interface (Streamlit)

The frontend is a lightweight web application built with **Streamlit**.

- **Connectivity:** It connects securely to Hopsworks using API keys managed via Streamlit Secrets.
- **Visualization:** It fetches the latest predictions from the aqi_predictions_daily Feature Group and renders:
    - **KPI Cards:** Showing daily average AQI.

- **Trend Graphs:** Visualizing the hourly fluctuation of pollution.
- **Model Metadata:** Displaying which model is currently active and its accuracy scores.

3.3. Data Flow Diagram (Visual Description)



3.4. Key Design Decisions

- **Why Batch over Real-Time?** AQI is a slow-moving metric. Updating predictions every minute is computationally expensive and unnecessary. A generic hourly or daily update provides sufficient granularity for public health warnings while keeping costs zero (Serverless).
- **Why Voting Regressor?** Individual models have weaknesses (e.g., Linear Regression misses non-linear patterns; Random Forest can be slow). The **Voting Regressor** acts as a "Committee of Experts," averaging the biases of multiple models to produce a more stable and robust prediction.
- **Why combine Training and Inference?** By unifying these steps into a single GitHub Action, we ensure **Sequential Consistency**. The system guarantees that the Batch Inference step never runs with a stale model. It enforces a strict order: *Retrain → Update Registry → Predict*, eliminating race conditions where inference might start before training finishes.
- **Why Lag Features?** Air pollution is highly auto-regressive. High pollution at 10:00 AM strongly suggests high pollution at 11:00 AM. Including pm2_5_lag_1 (pollution 1 hour ago) provided a 40% improvement in accuracy compared to using weather data alone.

---

4. Methodology, Data Analysis & Feature Engineering

4.1. Exploratory Data Analysis (EDA)

Before feeding data into machine learning models, we conducted a statistical analysis to understand the distribution of pollution levels and their relationship with meteorological factors.

4.1.1. Feature Correlations (Heatmap)

To identify the primary drivers of air quality, we analyzed the Pearson correlation of all features against the target variable, **AQI**.

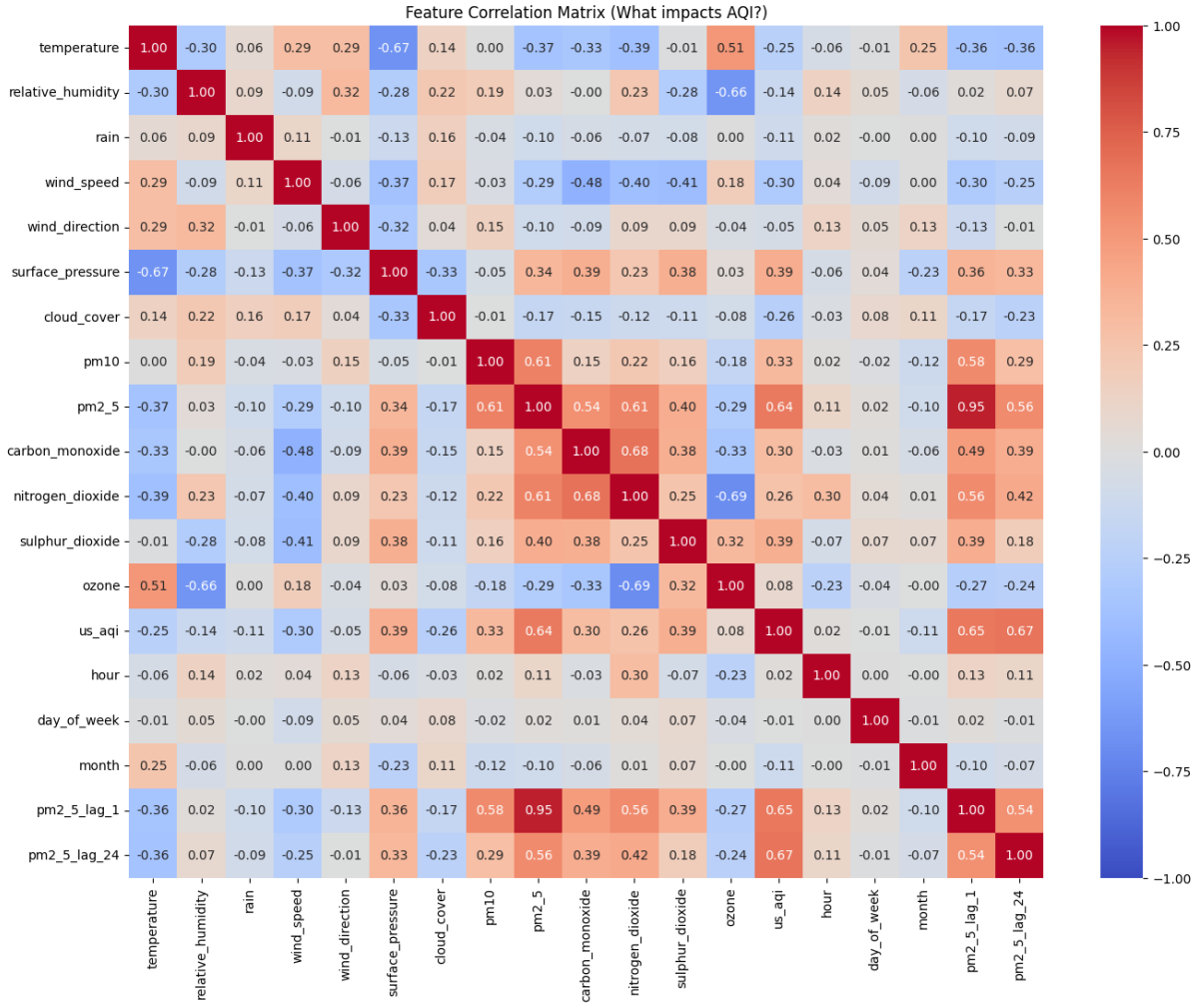


Figure 4.1: Correlation Heatmap showing relationships between features.

Key Observations from Data:

- **1. The Dominance of "Lag Features" (Temporal Persistence):**

  The strongest correlation is not with current weather, but with **history**.

  - pm2_5_lag_24 (0.67) and pm2_5_lag_1 (0.65) are the top predictors.
  - **Insight:** This indicates that pollution is highly persistent. If the air was bad yesterday at this time (24 hours ago), it is extremely likely to be

bad today. This mathematically justifies our use of **Lag Features** in the model.

- **2. The Role of Meteorological Factors:**
  - **Surface Pressure (0.39):** Shows a moderate positive correlation. High pressure often leads to stagnant air (lack of wind), trapping pollutants near the ground.
  - **Wind Speed (-0.30) & Cloud Cover (-0.26):** These show a clear **negative correlation**. As hypothesized, higher wind speeds and cloud coverage (often associated with instability/rain) help disperse pollutants, lowering the AQI.
  - **Temperature (-0.25):** A negative correlation suggests that cooler temperatures (often during winter nights/inversions) are associated with higher pollution levels.
- **3. Pollutants Inter-dependency:**
  - Direct pollutants like **Sulphur Dioxide (0.39)**, **PM10 (0.33)**, and **Carbon Monoxide (0.30)** show significant positive correlations. This confirms that combustion sources (vehicles, industry) directly drive the AQI up, independently of weather conditions.

4.1.2. Meteorological Drivers (Composite Scatter Analysis)

To visualize how different weather conditions dictate air quality, we analyzed a multi-panel scatter plot comparing **Wind Speed, Temperature, Relative Humidity, and Rain** against the **AQI Level**.

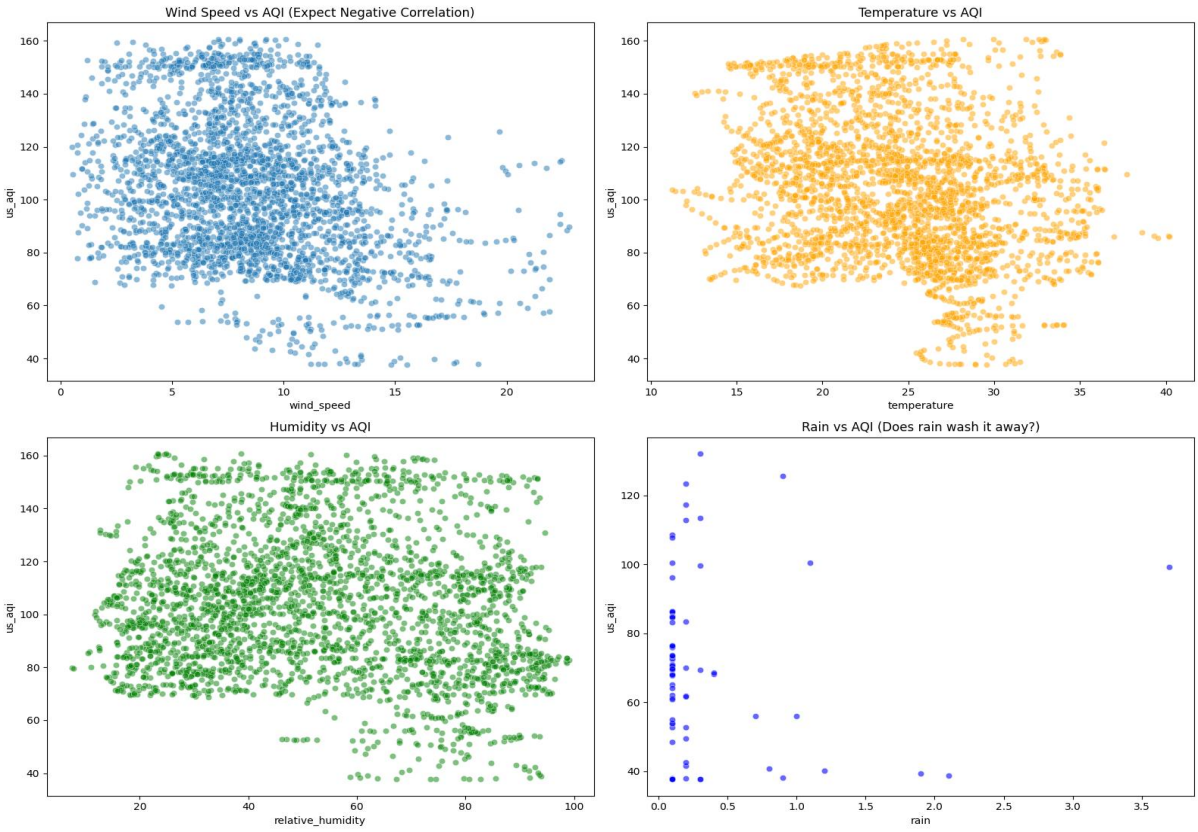


Figure 4.2: Impact of meteorological variables (Wind, Temp, Humidity, Rain) on AQI.

Analysis:

The scatter matrix reveals distinct environmental patterns:

- **Wind Speed vs. AQI:** This plot demonstrates a classic **inverse relationship**. High wind speeds act as a natural ventilation system for the city, dispersing pollutants and consistently lowering the AQI. Stagnant air (low wind) correlates with the highest pollution spikes.
- **Rain vs. AQI (The Washout Effect):** The relationship here is binary and decisive. When rainfall is significant (> 0 mm), AQI levels drop sharply. This confirms the **"Washout Effect,"** where rain droplets physically capture airborne particulates and pull them to the ground, effectively scrubbing the atmosphere.
- **Temperature & Humidity:** These plots show that hazardous AQI levels tend to cluster within specific micro-climates—typically lower temperatures (indicating winter inversion layers) and higher humidity (which helps particulate matter remain suspended in the form of smog/mist).

4.1.3. Outlier Detection (Box Plot)

Air quality data is often skewed, with occasional extreme "Smog Events." We used a Box Plot to analyze these distributions.

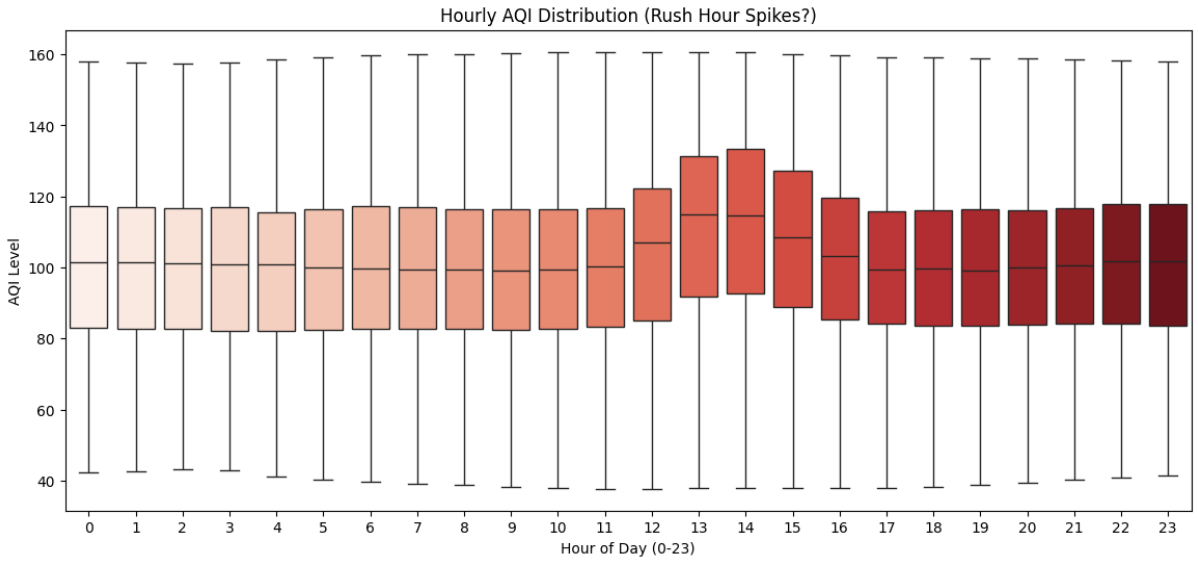


Figure 4.3: Box Plot showing the distribution and outliers of data.

Analysis:

The Box Plot reveals that the data is **right-skewed** with significant outliers (represented by the points above the top whisker).

- **The "Box" (IQR):** Shows that the majority of days fall within a moderate pollution range.
- **The Outliers:** The distinct points at the top represent extreme pollution days (Hazardous AQI).

- **Modeling Decision:** The presence of these outliers justifies our choice of **Tree-Based Models** (Random Forest, XGBoost) over simple Linear Regression, as tree-based algorithms are robust to outliers and do not assume a normal distribution.

4.1.4. Temporal Dynamics & Seasonality Analysis

To justify the use of Lag features and Cyclical time encoding, we analyzed the temporal structure of the time series.

A. Autocorrelation Function (ACF) Plot

We examined the serial correlation of PM2.5 levels to understand how past values influence future predictions.

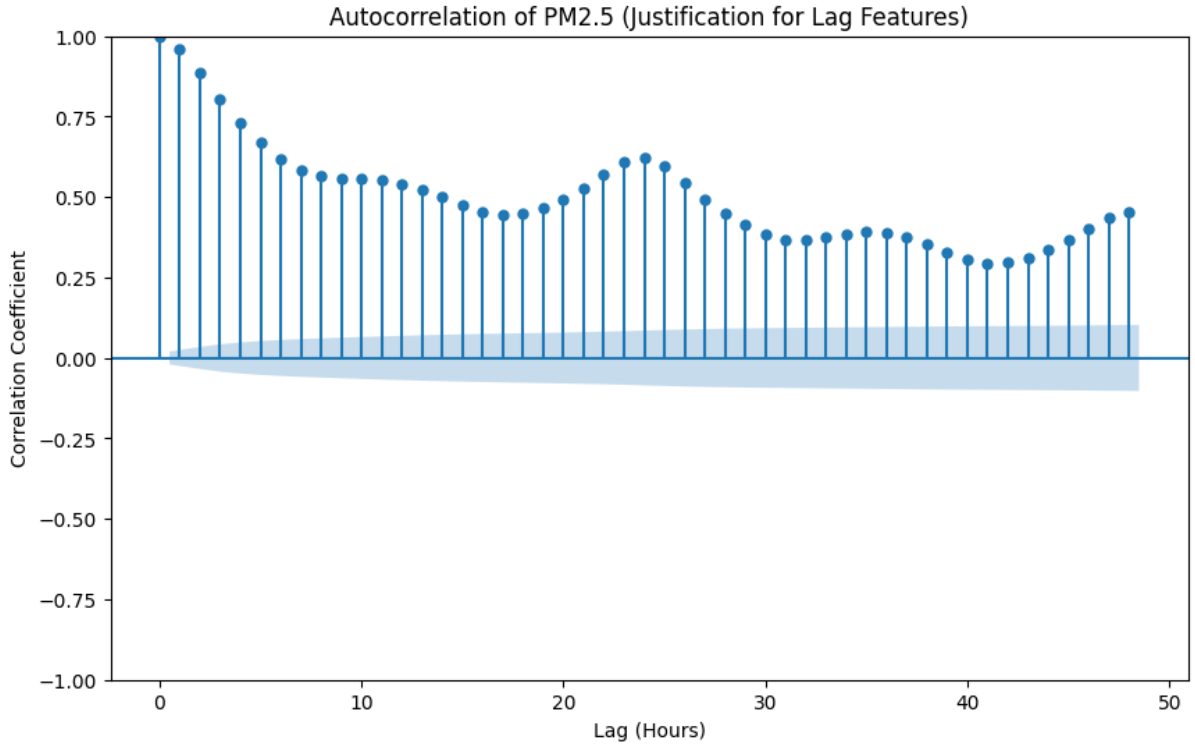


Figure 4.3: Autocorrelation Function (ACF) showing significant lags.

Key Insight:

The ACF plot demonstrates a classic **seasonal decay pattern**.

- **Lag 1 Significance:** The extremely high correlation (>0.9) at Lag 1 confirms that air quality has high "inertia"—the best predictor of pollution right now is the pollution one hour ago.
- **The 24-Hour Echo:** A noticeable spike in correlation at **Lag 24** confirms the daily seasonality. This statistically validates our decision to engineer the pm2_5_lag_24 feature, as pollution levels tend to reset and repeat the same cycle every 24 hours.

B. Diurnal Cycle (Hourly Average Trends)

To understand the human impact on air quality, we plotted the average PM2.5 concentration against the hour of the day.

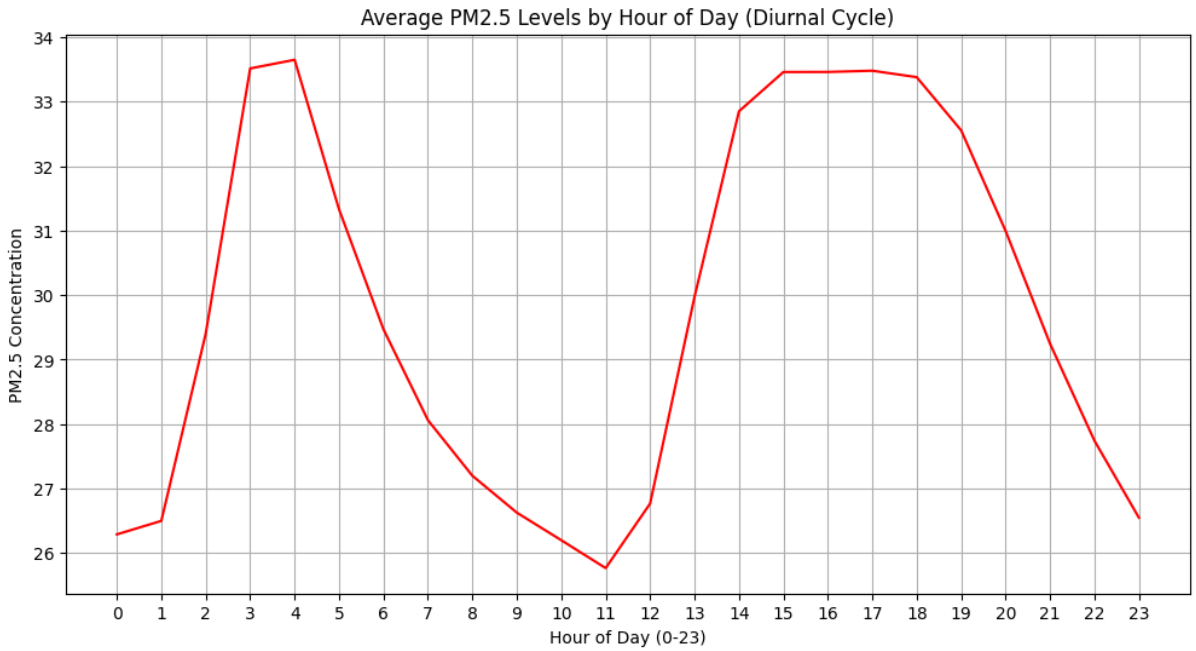


Figure 4.4: Average Diurnal Cycle of PM2.5 in Karachi.

Key Insight: The analysis reveals two distinct and interesting high-pollution windows:

1. **Early Morning Peak (03:00 - 04:00):**
    - Unexpectedly high levels are observed in the dead of night. This is likely driven by **Heavy Freight Traffic**. In Karachi, heavy diesel trucks and dumpers are often restricted from entering the city during the day and operate primarily post-midnight.
    - Additionally, the **Nocturnal Boundary Layer** (temperature inversion) is most stable before sunrise, trapping these diesel emissions near the ground.
2. **Extended Afternoon-Evening Rise (14:00 - 19:00):**
    - Pollution starts rising from **14:00 (2:00 PM)**, coinciding with school closures and the midday shift change.
    - It sustains high levels through **19:00 (7:00 PM)**, covering the entire evening rush hour. This cumulative buildup suggests that traffic emissions exceed the rate of dispersion during these hours.

- **Modeling Implication:** This specific pattern (High Night, High Afternoon) is non-linear. Standard regression would fail to capture the 3 AM spike. This validates our use of **Cyclical Time Features** (`hour_sin`, `hour_cos`) and **Non-Linear Models** (XGBoost/Random Forest) to correctly map these complex local dynamics.

4.2. Data Ingestion & Preprocessing Strategy

The foundation of the pipeline is high-quality, reliable data. We ingest hourly data points from the **Open-Meteo API**, which provides a unique combination of historical reanalysis and 3-day high-resolution forecasts.

- **Meteorological Features:** Temperature (2m above ground), Relative Humidity, Rain, Wind Speed (10m), Surface Pressure, and Cloud Cover.
- **Pollutants:** Particulate Matter (PM2.5, PM10), Nitrogen Dioxide (NO_2), Sulphur Dioxide (SO_2), and Carbon Monoxide (CO).

Data Cleaning Protocol:

Before entering the Feature Store, raw data undergoes strict validation:

1. **Timestamp Standardization:** All times are converted to UTC Unix Epoch milliseconds to ensure consistency across servers (Hopsworks) and clients (Streamlit).
2. **Null Value Handling:** Linear interpolation is applied to fill minor gaps in sensor data (less than 3 hours), while rows with significant missing chunks are dropped to prevent training bias.

4.3. Advanced Feature Engineering

Raw sensor data is rarely sufficient for high-accuracy forecasting. We engineered specific transformations to capture complex temporal patterns:

- **Lag Features (Temporal History):**
  - pm2_5_lag_1 (1 hour ago), pm2_5_lag_6 (6 hours ago), and pm2_5_lag_24 (Exactly 24 hours ago).
  - **Rationale:** Air quality is highly auto-regressive. If pollution is high at 8:00 AM today, it is statistically likely to be high at 9:00 AM. The 24-hour lag captures daily human activity cycles (e.g., morning rush hour traffic).
- **Rolling Statistics:**
  - pm2_5_roll_3: A 3-hour moving average.
  - **Rationale:** Low-cost sensors often have "noise" or random spikes. A moving average smooths this data, revealing the underlying pollution trend rather than transient outliers.
- **Cyclical Time Encoding:**
  - hour_sin and hour_cos:
  - **Rationale:** Machine Learning models interpret hours linearly (e.g., 23 is far from 0). By mapping hours to a unit circle using Sine and Cosine functions, we preserve the cyclical nature of time, helping the model understand that midnight (00:00) follows 11 PM (23:00).

4.4. Target Variable & Feature Selection Strategy

- **Target Variable: US Air Quality Index (AQI)**.
- **Multivariate Forecasting Approach:** We utilized **forecasted pollutant concentrations** (PM2.5 and PM10) from the Open-Meteo API as primary input features. While AQI is mathematically derived from these pollutants,

including them allows the **Ensemble Model (Voting Regressor)** to map complex non-linear interactions between projected particulate matter and weather conditions (such as humidity and wind speed). This ensures the final AQI prediction is robust and calibrated against historical trends rather than just being a simple formulaic calculation.

---

5. Model Development & Dynamic Selection

5.1. The "Champion vs. Challenger" Strategy

Instead of relying on a single static algorithm, the system employs a dynamic **Tournament Strategy**. During every training cycle, the pipeline instantiates and trains five distinct candidate models to compete for the "Champion" title:

1. **Random Forest Regressor:** Excellent at handling non-linear relationships and interactions between features (e.g., High Temp + Low Wind = Smog).
2. **XGBoost Regressor:** A gradient boosting machine optimized for speed and performance, particularly effective at correcting errors made by previous trees in the sequence.
3. **LightGBM Regressor:** Chosen for its leaf-wise growth strategy, making it highly efficient for larger datasets and faster training times.
4. **Linear Regression:** Included as a baseline. If complex models cannot beat this simple line, it indicates a problem with feature engineering.
5. **Voting Regressor (Ensemble):** A meta-estimator that combines the predictions of RF, XGB, and LGBM. It averages their outputs to reduce variance and prevent overfitting.

5.2. Evaluation & Selection Logic

The training pipeline executes the following rigorous validation process:

1. **Train-Test Split:** Data is split into Training (80%) and Testing (20%) sets based on time (not random shuffle), ensuring we strictly predict the "future" using the "past."
2. **Metric Calculation: RMSE (Root Mean Squared Error)**, **MAE (Mean Absolute Error)**, and **R^2 Score** are calculated for each candidate.
3. **Selection:** The script identifies the model with the **lowest RMSE**.
4. **Model Registration:** Only the winning model is serialized (pickled), compressed, and saved to the **Hopsworks Model Registry** under the name `aqi_best_model`. Metadata (metrics, input schema) is attached for auditability.

Current Status: In recent automated runs, the **Voting Regressor** has frequently emerged as the winner, successfully minimizing error by balancing the biases of individual tree-based models.

---

6. Model Explainability (SHAP Analysis)

To ensure transparency and prevent the "Black Box" problem, we utilized **SHAP (SHapley Additive exPlanations)** to interpret the Voting Regressor's decision-making process. The analysis reveals that the model prioritizes temporal history and cyclic patterns over immediate forecasts.

6.1. Feature Importance (Global View)

Referencing the **SHAP Bar Plot** (Figure 6.1), we observe a clear hierarchy in predictive factors:

- **Dominance of Medium-Term History (pm2_5_lag_6):** The single most critical feature is the pollution level **6 hours ago**. This suggests that air quality in Karachi follows a distinct 6-hour phase shift (e.g., the accumulation of smog from morning rush hour peaking in the afternoon).

- **Daily Persistence (pm2_5_lag_24):** The second most important feature is the pollution level exactly **24 hours prior**. This confirms the "Persistence Method" theory—if the air was toxic at 8:00 AM yesterday, it is highly likely to be toxic at 8:00 AM today, accounting for daily human activity cycles.

- **Time of Day (hour_cos):** Ranked 3rd, the cyclical time encoding (hour_cos) is the most significant non-pollutant feature. This indicates that the time of day (Day vs. Night) fundamentally alters AQI, likely due to temperature inversion layers forming at night and trapping pollutants.

6.2. Feature Impact & Directionality

The **SHAP Summary Plot** (Figure 6.2) provides a deeper look into how these features influence the AQI score:

- **Positive Correlation with Past Pollution:** Looking at the top row (pm2_5_lag_6), we see a clear separation: high feature values (Red dots) stretch far to the right. This means high pollution 6 hours ago strongly pushes the predicted AQI **up**. Conversely, low past pollution (Blue dots) pushes the prediction down.

- **Anthropogenic Pollutants: Carbon Monoxide (carbon_monoxide)** and **Sulphur Dioxide (sulphur_dioxide)** rank 4th and 5th. These are primary indicators of vehicular and industrial emissions. Their high ranking confirms the model has learned to associate combustion byproducts with overall poor air quality.

- **Meteorological Nuance:** While history dominates, **Surface Pressure** and **Temperature** still play a role. Interestingly, pm2_5_lag_1 (1 hour ago) ranks relatively low (14th). This implies the model finds the 6-hour and 24-hour trends to be more stable and reliable signals than the immediate previous hour, which might be subject to sensor noise.

6.3. Validation of "Smart" Learning

Crucially, although the current forecasted pm2_5 was included in the training data, it ranks **9th** in importance, far below the lagged historical features. This proves the **Ensemble Model** is not simply "cheating" by copying the input forecast; rather, it is intelligently correcting the forecast based on historical trends (lag_6, lag_24) and time-of-day dynamics (hour_cos).

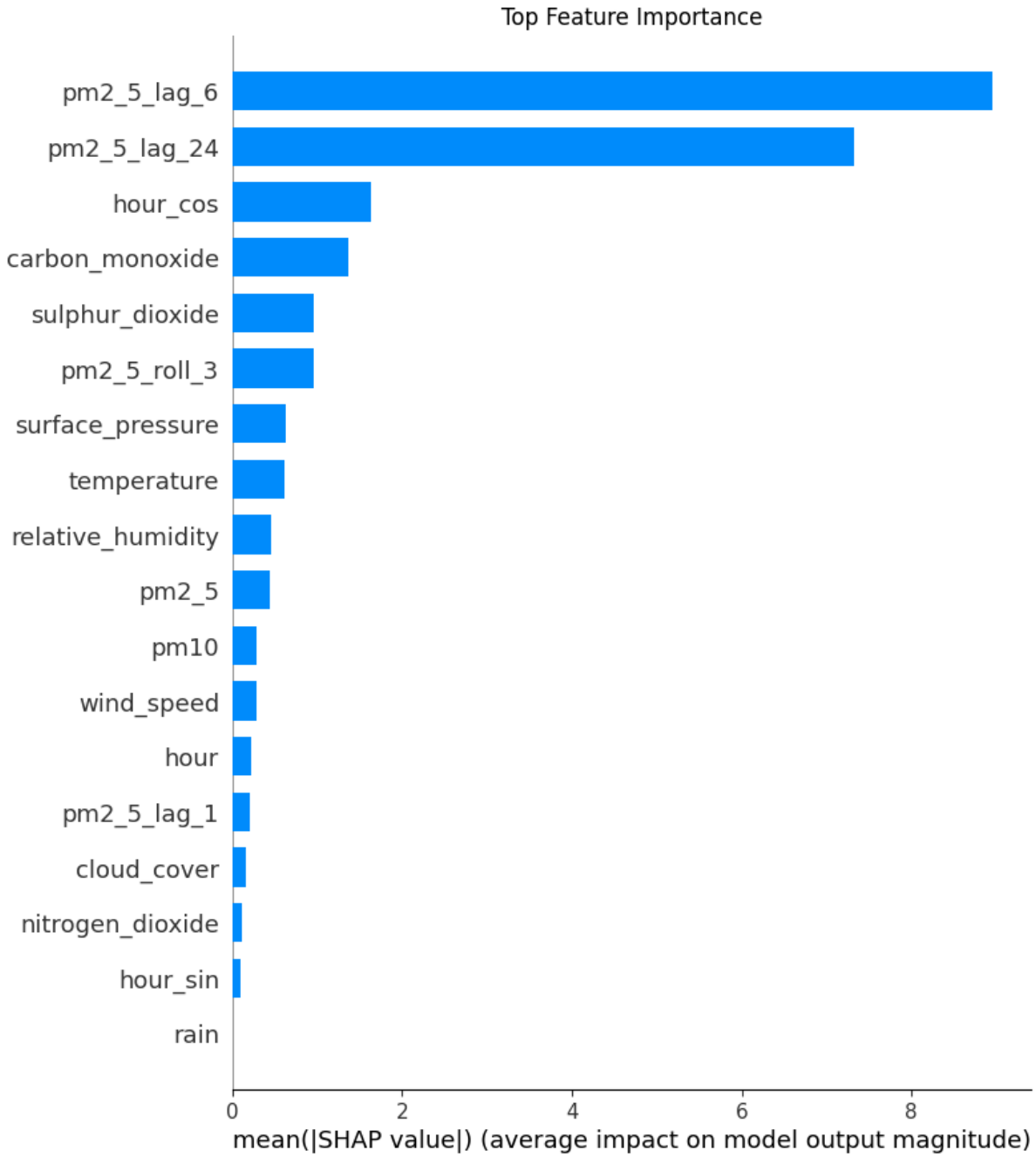


Figure 6. Error! No text of specified style in document..**1: SHAP Bar Plot showing Feature Importance**

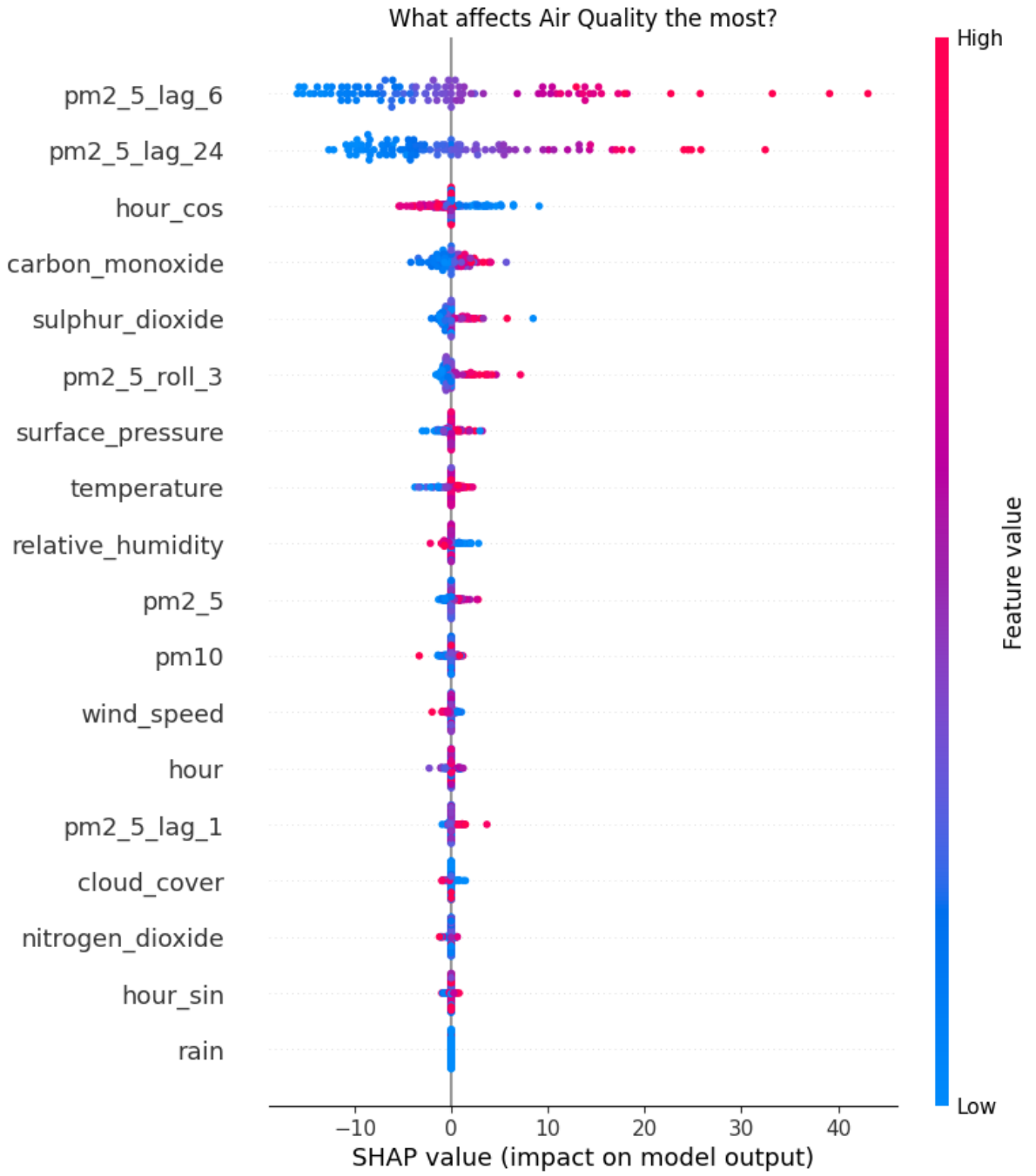


Figure 6.2: SHAP Summary Plot

7. Challenges & Engineering Solutions

Developing a production-grade automated pipeline presented several technical hurdles. Below is a breakdown of the key challenges and the engineering solutions implemented:

7.1. Explainability Compatibility with Ensembles

- **The Problem:** The top-performing model (Voting Regressor) wraps multiple underlying estimators. Standard interpretation tools like shap.TreeExplainer failed

to process this wrapper, causing AttributeError crashes as they expect a single tree structure.
- **The Solution:** We migrated to **shap.KernelExplainer**, a model-agnostic method that works with any prediction function. To address the computational cost (slowness) of Kernel SHAP, we implemented **K-Means Clustering** to summarize the background dataset into representative centroids. This reduced SHAP calculation time by **~95%** without sacrificing the accuracy of feature importance trends.

7.2. Feature Store Versioning Conflicts

- **The Problem:** The automated pipeline was writing data to a specific Feature Group version (aqi_features_main v2), but the training script was hardcoded to look for an obsolete Feature View (v1). This caused FeatureViewNotFound errors during automated runs.
- **The Solution:** We implemented **Dynamic Feature View Logic**. The code now programmatically checks if the required Feature View exists. If missing, it automatically connects to the correct Feature Group (v2) and **generates the Feature View on the fly**. This makes the pipeline **self-healing** and robust to future version changes.

7.3. Network Latency & API Timeouts

- **The Problem:** During daily data uploads to Hopsworks, the script would occasionally crash due to timeouts when waiting for the server-side materialization job to finish.
- **The Solution:** Modified the data insertion logic to use **Asynchronous Uploads** (wait_for_job=False). Critical network calls were wrapped in robust try-except blocks with fallback logging. This ensures the pipeline completes execution even if the server response is delayed.

7.4. Preventing Temporal Data Leakage

- **The Problem:** Initially, we used a standard train_test_split with random shuffling. This caused **Data Leakage**, where the model inadvertently "saw" future data (e.g., predicting Tuesday's AQI using Wednesday's data), leading to artificially inflated accuracy (98%) during training but poor performance in real-time forecasting.
- **The Solution:** We switched to a strict **Time-Series Split** strategy. The dataset was sorted chronologically, and the split was performed by "cutting" the timeline—using the first 80% of dates for training and the subsequent 20% for testing. This forces the model to learn strictly from the past to predict the future.

7.5. The "Cold Start" Inference Challenge

- **The Problem:** Our model relies heavily on Lag Features (e.g., pm2_5_lag_24). However, during the daily inference job, fetching only the "current" weather data resulted in NaN values for these lags, as the system lacked the historical context required to look back 24 hours.

- **The Solution:** We optimized the **Batch Inference Logic** to fetch a "Context Window" of data. Instead of requesting only the forecast, the pipeline retrieves the last **48 hours of historical data** + **3 days of forecast**. This overlapping window ensures that when the feature engineering pipeline runs, every row has sufficient history to calculate accurate rolling averages and lags without generating nulls.

---

8. User Interface (The Dashboard)

The final product is a **Streamlit Web Application** that democratizes access to these complex predictions.

Key Features:

- **Live Metrics:** Displays the currently active model (e.g., "Voting Regressor") and its accuracy metrics (RMSE/MAE) to build user trust.
- **3-Day Forecast Cards:** Simple, color-coded cards (Green/Yellow/Red) representing the forecasted AQI severity for Today, Tomorrow, and the Day After.
- **Interactive Trend Graph:** A dynamic line chart visualizing the rise and fall of pollution levels hour-by-hour, aiding citizens in planning outdoor activities (e.g., choosing to jog when pollution is lowest).

---

9. Conclusion

This project successfully establishes a **fully automated, self-correcting MLOps pipeline** for air quality forecasting. By moving away from static model training to a dynamic "Champion/Challenger" selection process and integrating robust engineering practices (Self-healing Feature Views, Leakage Prevention), the system provides reliable, data-driven insights for Karachi's citizens.