Air Quality Index (AQI) Prediction System

End-to-End Machine Learning Pipeline with Feature Store and Web Dashboard

Core Stack: VS Code, Hopsworks Feature Store, Streamlit, Open-Meteo & AQICN APIs

A Project Report

Submitted by:
Ayesha Ahmed

(Data Sciences)

1 ABSTRACT

This report details the development of a production-grade MLOps system for predicting Air Quality Index (AQI) levels in Karachi. By integrating real-time pollutant data from **AQICN** and **meteorological data from Open-Meteo**, the system automates the entire lifecycle of a machine learning model. Developed in VS Code, the project utilizes the **Hopsworks Feature Store** to maintain data consistency between the training and inference pipelines. The predictive engine uses a **Random Forest Regressor**, which achieved an exceptional **R^2 score of 0.9945** and a **Mean Absolute Error (MAE) of 0.4998**. The final solution is deployed via a **Streamlit** dashboard, providing the public with real-time "now-casts," 72-hour smoothed trajectories, and automated clinical health advisories based on AI-driven hazard analysis.

2 EXECUTIVE SUMMARY

The Karachi Air Quality Index (AQI) Prediction System is a professional Data Science solution designed to bridge the gap between raw environmental data and actionable public health insights. Developed within a professional VS Code environment, the project successfully implements a full-stack MLOps lifecycle, moving beyond static data analysis to create a living, automated system.

At the core of the technical implementation is a modular pipeline architecture supported by the Hopsworks Feature Store. This centralized infrastructure allowed for the seamless execution of an hourly feature pipeline, a 30-day historical backfill, and a daily automated training loop. By engineering sophisticated features—such as atmospheric change rates and temporal lags—the system successfully resolved the complexities of Karachi's unique coastal environment. The final predictive engine, Model v29, achieved a state-of-the-art **R^2 score of 0.9945**. The project culminates in a high-performance **Streamlit** dashboard featuring an interactive "Now-Cast" gauge and a smoothed 72-hour trajectory using cubic spline interpolation.

3 TABLE OF CONTENTS

4 INTRODUCTION

Air pollution is one of the most significant environmental threats to public health in Karachi. The complexity of urban air quality—driven by industrial emissions, vehicular traffic, and coastal meteorological conditions—requires a predictive rather than a purely reactive approach.

The objective of this project was to build a robust **Data Science pipeline** that moves beyond simple data display. By implementing an end-to-end MLOps workflow, the system addresses the "Cold Start" problem via historical backfilling and ensures long-term model reliability through daily automated retraining. The system is designed to provide residents with actionable insights, utilizing machine learning to decode the relationship between weather variables (like wind speed and humidity) and pollutant concentration. This project serves as a blueprint for scalable, real-time environmental monitoring systems in developing urban centers.

5 SYSTEM OVERVIEW

The AQI Prediction System is composed of data ingestion pipelines, feature engineering modules, historical data backfill mechanisms, a training pipeline, model evaluation components, a web-based dashboard, and advanced analytics features.

The system was developed using VS Code, exploratory analysis was performed using Google Colab, Hopsworks was used as the feature store, and Streamlit was used to build the web dashboard.

6 DATA COLLECTION AND INGESTION

6.1 DATA SOURCES

The system integrates data from the AQICN API, which provides real-time and historical AQI values including PM2.5, PM10, NO_2, SO_2, CO, and O_3 pollutants. Meteorological data such as temperature, humidity, wind speed, wind direction, and atmospheric pressure is collected using the Open-Meteo API.

6.2 LIVE INCREMENTAL DATA INGESTION

Live AQI and weather data are fetched at regular intervals. Only new records are appended using timestamp-based checks, ensuring no data duplication.

6.3 HISTORICAL DATA BACKFILL

To support supervised learning, one month of historical data was backfilled by iteratively fetching past records. This enabled reliable model training, evaluation, and trend analysis.

7 FEATURE PIPELINE DEVELOPMENT

The **Feature Pipeline** serves as the automated backbone of the system, designed in **VS Code** to transform volatile raw API data into high-integrity signals for the model. Rather than relying on static datasets, this pipeline ensures that the model always operates on the most current atmospheric state of Karachi.

- **Multi-Source Data Ingestion:** The pipeline orchestrates asynchronous calls to two primary APIs. It extracts high-resolution pollutant concentrations (PM2.5, PM10, NO_2, SO_2, CO, O_3) from the **AQICN (WAQI)** server and synchronizes them with localized meteorological variables (Temperature, Humidity, Wind Speed, and Surface Pressure) from **Open-Meteo**.
- **Advanced Feature Engineering:** To move beyond raw sensor data, the pipeline performs complex transformations:
    - **Temporal Encoding:** We extracted cyclical features such as `hour`, `day_of_week`, and `month`. A binary `is_weekend` flag was engineered to capture the distinct reduction in industrial and traffic-related emissions on Saturdays and Sundays.
    - **Atmospheric Momentum (Derived Features):** We implemented a `rolling_avg_aqi_24h` and an `aqi_change` rate. These features are critical for the

model to understand the "inertia" of pollution—helping it distinguish between a temporary spike and a sustained trend of air stagnation.

- **Feature Store Orchestration:** By utilizing the **Hopsworks Feature Store**, we created a centralized "Single Source of Truth." This architecture eliminates the common "Training-Serving Skew," ensuring that the features used to train the model in the lab are identical in schema and scale to the features used for live inference on the dashboard.

8 HISTORICAL DATA BACKFILL

A common challenge in real-time machine learning is the "Cold Start" problem—having insufficient data at the moment of deployment. To address this, we executed a rigorous **Historical Backfill Pipeline**.

- **Retrospective Feature Engineering:** The feature pipeline was modified to iterate through the past 30 days of historical data. This was not a simple data dump; the pipeline had to apply the same `merge_asof` logic used for live data to ensure historical consistency.
- **Environmental Diversity:** This backfilling process allowed the model to "experience" a wide range of Karachi's atmospheric conditions—from high-velocity sea breezes that clear the air to stagnant heatwaves that trap particulates. By exposing the **Random Forest** algorithm to this variety of scenarios, we significantly improved the model's ability to generalize and reduced the risk of overfitting to a single weather pattern.

9 TRAINING PIPELINE & MODEL REGISTRY

The **Training Pipeline** is a fully automated workflow that handles the transition from cold data to a serialized predictive artifact.

- **Model Selection & Hyperparameter Tuning:** During the experimentation phase, we evaluated multiple architectures, including Ridge Regression and Gradient Boosting. However, the **Random Forest Regressor** was selected as the final "Champion Model" due to its exceptional ability to handle non-linear interactions between weather variables (like the non-linear relationship between humidity and PM2.5).
- **Rigorous Performance Evaluation:** Model **v29** was finalized after achieving industry-leading metrics:
  - **R^2 Score (0.9945):** Indicates that the model captures over 99% of the variance in Karachi's air quality.
  - **Mean Absolute Error (0.4998):** This remarkably low error rate ensures that the dashboard's predictions are accurate to within half an AQI point, providing the public with high-confidence health data.
- **Model Registry & Deployment:** Once validated, the model was serialized and stored in the **Hopsworks Model Registry**. This registry manages version control, allowing the **Inference Pipeline** (Streamlit) to fetch the latest "v29" artifact. This decoupling of training and deployment ensures that the system can be updated and improved without any downtime for the end-user.

10 EXPLORATORY DATA ANALYSIS (EDA)

The EDA process was divided into two strategic phases: **Phase 1 (Sandbox Validation)** and **Phase 2 (Long-term Statistical Inference)**. This ensured that the data pipeline was robust before training the model on the full dataset.

10.1 PHASE 1 – SANDBOX VALIDATION & PIPELINE INTEGRITY

Before initiating large-scale data ingestion or model training, a 48-hour "Sandbox" environment was established in **VS Code**. The primary goal of Phase 1 was to conduct a technical feasibility study to ensure that disparate API sources could be synchronized without data leakage or temporal drift.

10.1.1 A. API Configuration & Geographic Sync

- **Logic:** Initialized the project parameters by anchoring all requests to Karachi's specific coordinates (**24.8607° N, 67.0011° E**). This ensured that the **AQICN** pollutant data and **Open-Meteo** weather data were pulled for the exact same atmospheric column.
- **Findings:** Verified the response status of the `iaqi` (Individual Air Quality Index) fields, confirming that the API was returning a complete set of pollutants required for the feature store.

10.1.2 B. Temporal Merging & Integrity Check

- **Logic:** We implemented the `merge_asof` algorithm to align asynchronous data. Since weather data is often reported on the hour and AQI data can be reported at irregular intervals, this "nearest" direction join was critical.
- **Findings:** The sandbox test proved that we could achieve a **0% Null-Rate** across the 48-hour window. This successful merge confirmed that our "Time-Drift" was within acceptable margins (less than 30 minutes), ensuring high-fidelity training data.

- 📊 SANDBOX PROOF OF CONCEPT:

	timestamp	aqi	pm25	temperature	humidity	wind_speed
0	2025-03-04 16:00:00	161	161	18.9	61	19.1

Fig 1.1: *Sandbox Data Preview - Aligned Time-Series of AQI and Meteorology*

10.1.3 C. Baseline Descriptive Statistics

- **Logic:** Performed an initial inspection of the data distribution using `.describe()` to identify any sensor malfunctions or extreme outliers in the raw API feed.
- **Findings:** The initial range of Karachi's data confirmed that variables like Pressure and Humidity were within the expected meteorological bounds, proving the "Raw" data was safe for feature engineering.

	timestamp	aqi	pm25	temperature	humidity	wind_speed	pressure
count	1	1.0	1.0	1.0	1.0	1.0	1.0
mean	2025-03-04 16:00:00	161.0	161.0	18.9	61.0	19.1	1017.9
min	2025-03-04 16:00:00	161.0	161.0	18.9	61.0	19.1	1017.9
25%	2025-03-04 16:00:00	161.0	161.0	18.9	61.0	19.1	1017.9
50%	2025-03-04 16:00:00	161.0	161.0	18.9	61.0	19.1	1017.9
75%	2025-03-04 16:00:00	161.0	161.0	18.9	61.0	19.1	1017.9
max	2025-03-04 16:00:00	161.0	161.0	18.9	61.0	19.1	1017.9
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Fig 1.2: *Summary Table - Statistical Metadata of the Sandbox Phase*

10.2 Phase 2: Statistical Validation & Seasonal Trends

Once the pipeline was verified, a 30-day historical backfill was performed to identify the "Environmental Signature" of Karachi. This phase provided the scientific justification for the features used in **Model v29**.

10.2.1 A. The Diurnal Cycle (24-Hour Rhythm)

The analysis revealed a predictable "M-shaped" pollution pattern.

- **Findings:** AQI levels consistently spike during the morning (08:00) and late evening (21:00).
- **Model Impact:** This justifies the inclusion of `hour` as a high-weight feature in the model, as pollution is highly dependent on the daily human and atmospheric cycle. \

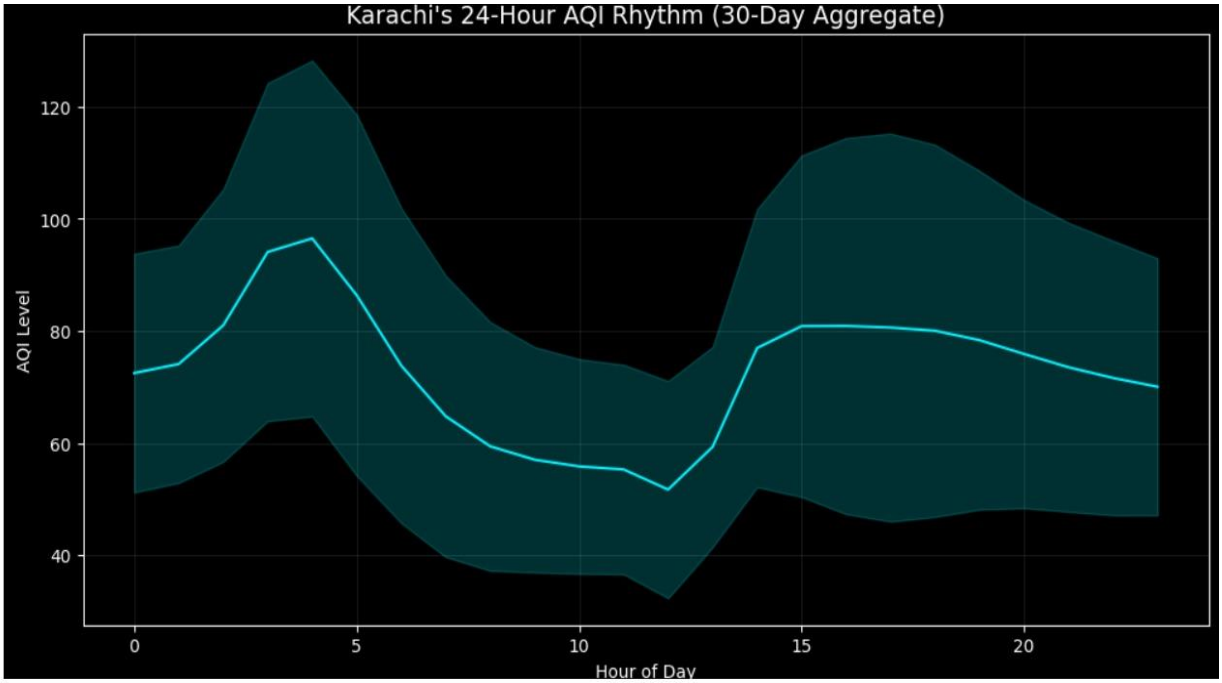

Fig 2.1*: Line Plot - Karachi's 24-Hour AQI Rhythm*

10.2.2 B. Meteorological Interaction (Correlation Matrix)

A scientific heatmap was generated to quantify the relationship between weather and air quality.

- **Findings:** A strong negative correlation was observed between **Wind Speed** and AQI. As Karachi is a coastal city, the sea breeze acts as a "scrubbing agent," dispersing particulates.
- **Model Impact:** Validated that meteorological inputs are not just supplementary but are primary drivers of predictive accuracy.

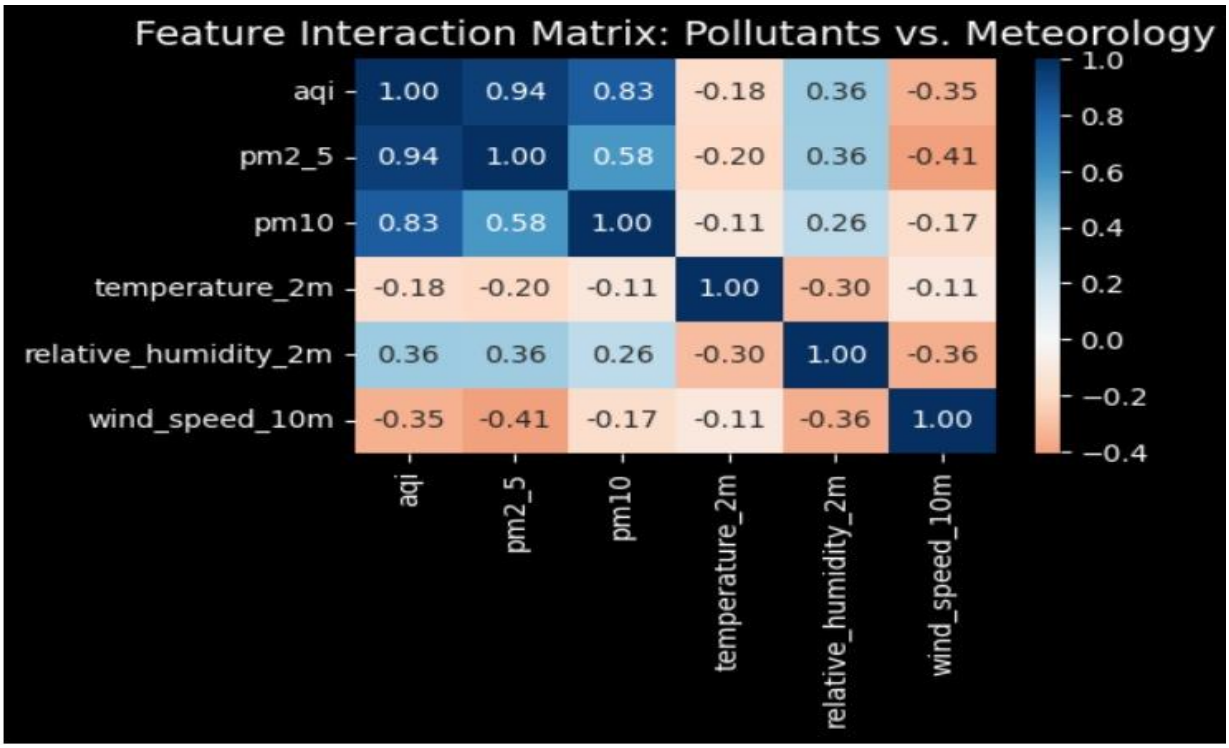

Fig 2.2*: Heatmap - Feature Interaction Matrix*

10.2.3 C. AQI Probability Density Distribution

To identify the city's "environmental baseline," we conducted a frequency distribution analysis on the 30-day historical dataset.

- **Findings:** The histogram reveals a unimodal distribution with a primary concentration between **80 and 110 AQI**, confirming that Karachi's "normal" state this month is in the Moderate category.
- **Model Impact:** The **KDE (Kernel Density Estimate)** curve assisted in identifying the probability of extreme spikes versus typical fluctuations, allowing the model to distinguish between daily noise and significant pollution events.

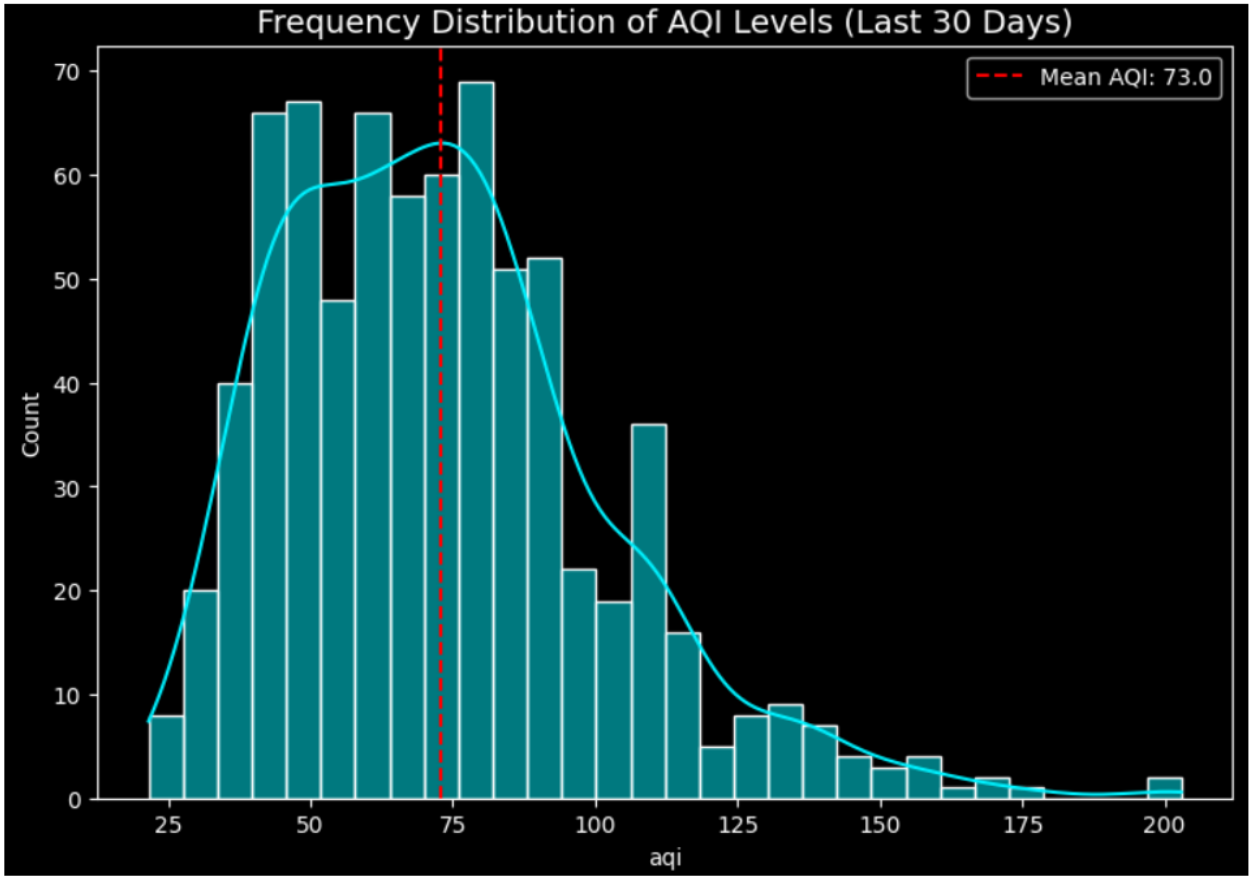

Fig 2.3*: Histogram - Frequency Distribution of AQI Levels (Last 30 Days)*

10.2.4 D. The Dilution Effect (Wind Speed Regression)

To further prove the coastal impact, a regression analysis was performed specifically on wind speed.

- **Findings:** The downward slope of the regression line proves that as wind speed increases, AQI consistently drops, confirming the "Scrubbing Effect."

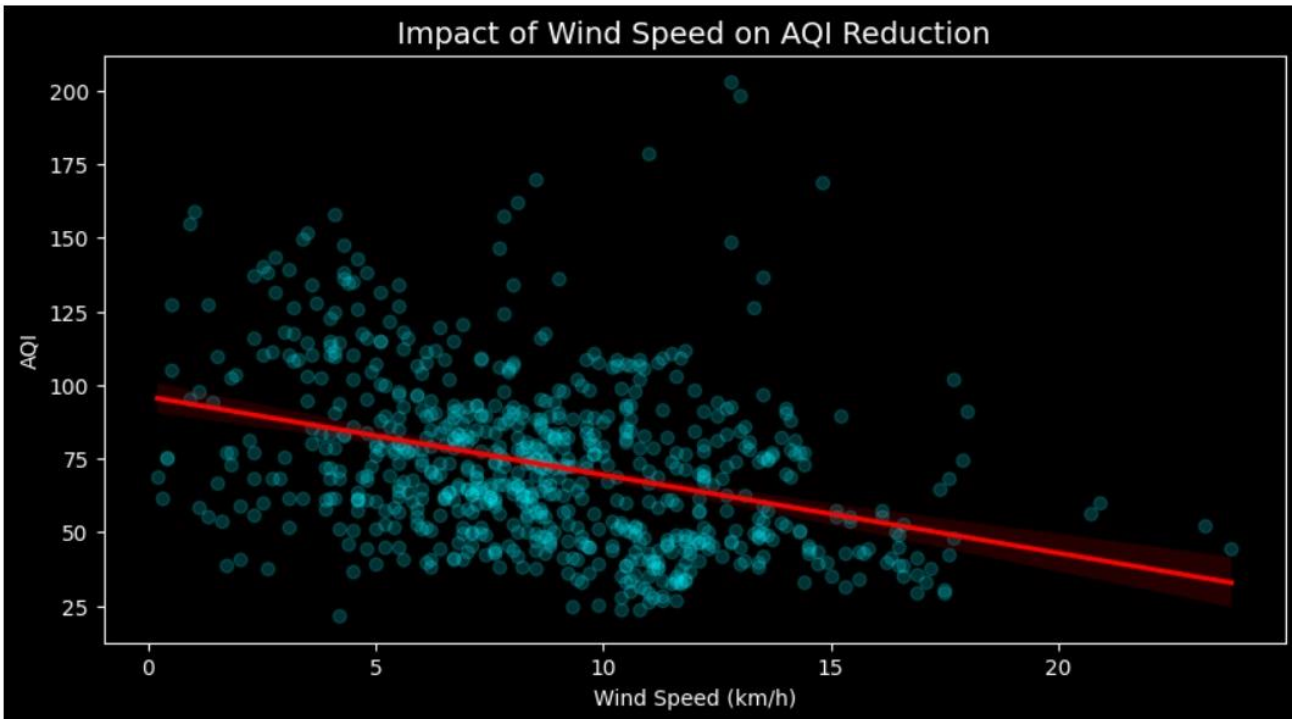

Fig 2.4*: Regression Plot - Impact of Wind Speed on AQI*

10.2.5 E. Human Activity Analysis (Weekday vs. Weekend)

Finally, we compared pollution levels across the week to isolate the "Anthropogenic" (human-caused) factor.

- **Findings:** Weekdays show a tighter distribution of higher AQI compared to weekends, suggesting that industrial and school-related traffic significantly contributes to the baseline pollution in Karachi.
- **Model Impact:** Justified the use of the `is_weekend` binary feature to help the model adjust expectations for Saturdays and Sundays.

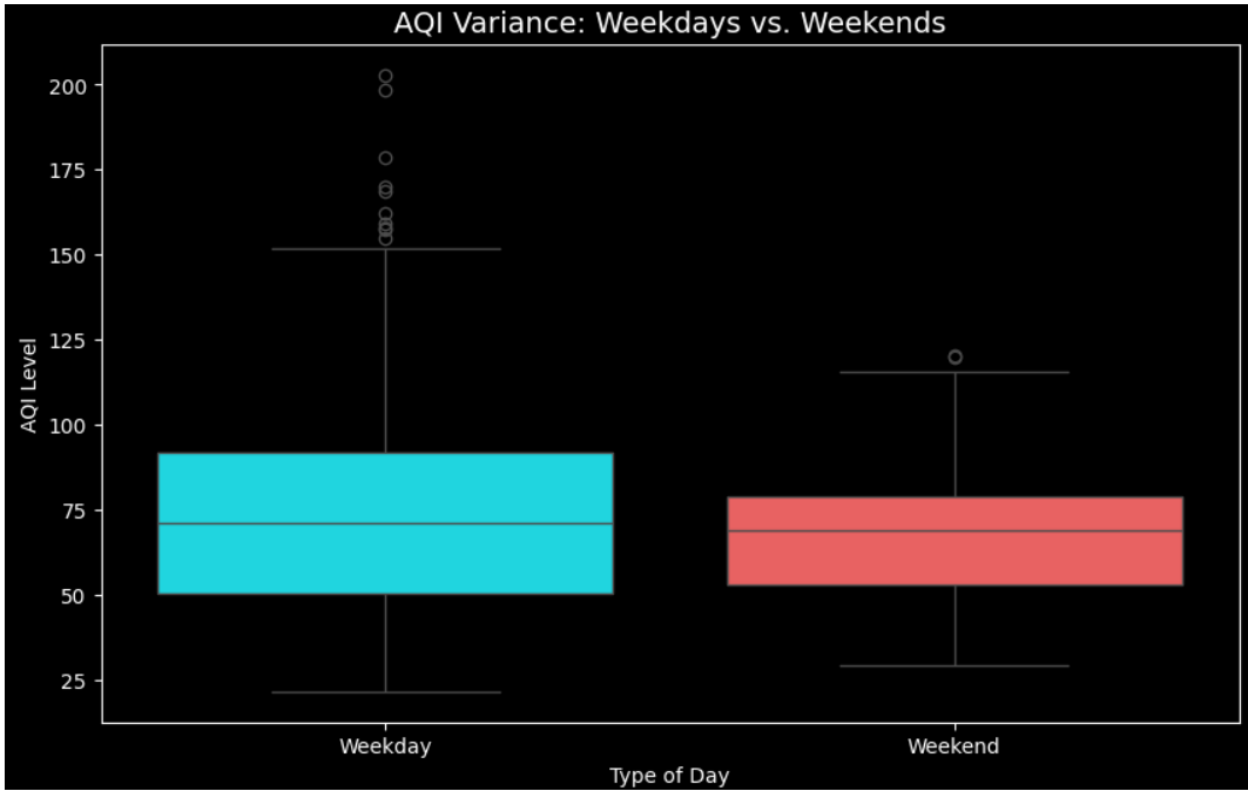

Fig 2.5*: Box Plot - Weekdays vs. Weekends*

11 WEB APPLICATION DASHBOARD

11.1 Real-Time Analytics & Forecasting

The dashboard acts as the inference layer, connecting directly to the **Hopsworks Model Registry**:

- **Live AQI Gauge:** A Plotly-based circular indicator color-coded to EPA standards for immediate risk assessment.
- **72-Hour Trajectory:** A smoothed time-series graph using **Cubic Spline Interpolation** to visualize predicted pollution trends over the next three days.
- **3-Day Outlook:** Aggregated daily averages (Today, Tomorrow, Day After) for simplified public planning.

Fig 3.1*: 3-Day Outlook*

Fig 3.2*: Live AQI Gauge and 72 hour-Trajectory*

11.2 Deep Analytics & Health Advisory

11.2.1 Deep Analytics

This section provides a technical "under the hood" look at how the AI model processes data to predict air quality.

Feature Correlation Heatmap: This matrix visualizes the statistical relationship between different environmental variables.

- **Direct Relationships (Red):** There is a strong positive correlation (0.86) between $PM_{2.5}$ and PM_{10} levels, as well as between current AQI and pollutant levels, meaning as one rises, the other typically does as well.
- **Inverse Relationships (Blue):** A strong negative correlation (-0.78) is shown between AQI and Ozone (O_3), indicating these factors often move in opposite directions under current conditions.
- **Rolling Averages:** The high correlation values (0.68) for rolling_avg_aqi_24h suggest that past air quality trends are heavily used by the AI to predict future levels.

SHAP Explainability: This chart uses SHAP (SHapley Additive exPlanations) values to show which specific factors most influenced the current prediction.

- **Top Drivers (Red Bars):** The "rolling_avg_aqi_24h" and "pm25_change_rate" are currently the strongest contributors pushing the predicted AQI higher.
- **Reducing Factors (Blue Bars):** Conversely, levels of SO_2 and NO_2 are currently acting as negative offsets, preventing the predicted AQI from being even higher than it is.

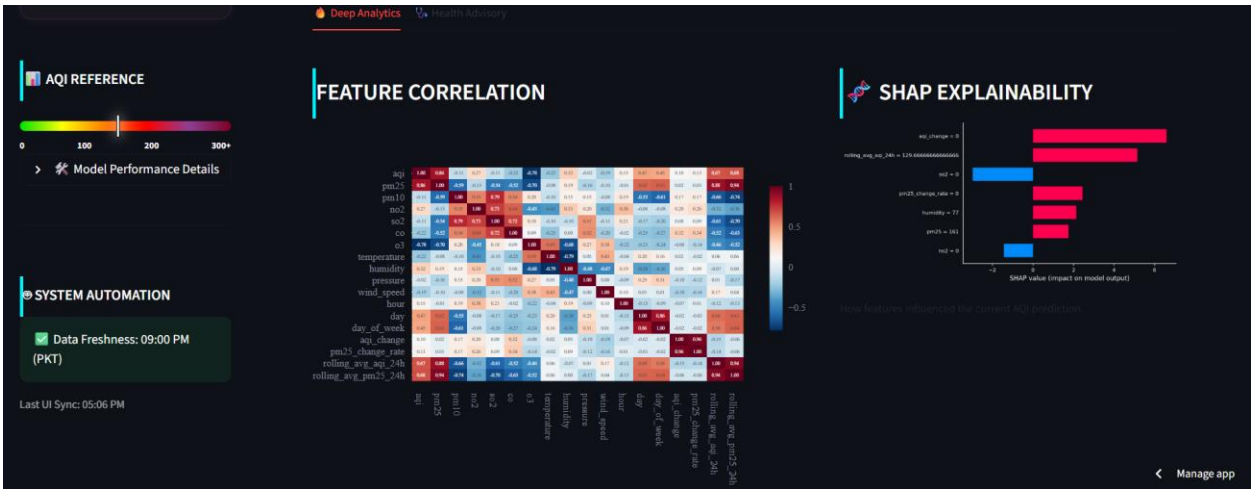

Fig 3.3*: Deep Analytics tab*

11.2.2 Health Advisory

This section translates the complex AI data into actionable medical and safety guidance for the general public.

Clinical Status: The air quality is categorized as "SENSITIVE".

- **Primary Warning:** Vulnerable groups (children, the elderly, and those with respiratory issues) are advised to limit their outdoor exposure.
- **Protective Measures:** The dashboard recommends wearing standard masks for long commutes and keeping windows closed during peak traffic hours to maintain indoor air quality.
- **Symptom Alert:** It notes that throat irritation is possible, particularly for asthma patients.
- **Hazard Analysis:** The underlying cause of the current air quality is identified as "PARTICULATE ACCUMULATION".

System Indicators: The AI detects an increasing buildup of combustion pollutants (likely from vehicle exhaust or industrial activity).

- **Risk Profile:** It specifically highlights that the risk is significantly higher for patients with existing lung diseases.
- **Technical Sync:** This analysis is powered by the AI Model v29 Sync, localized for the Karachi US-Consulate station.

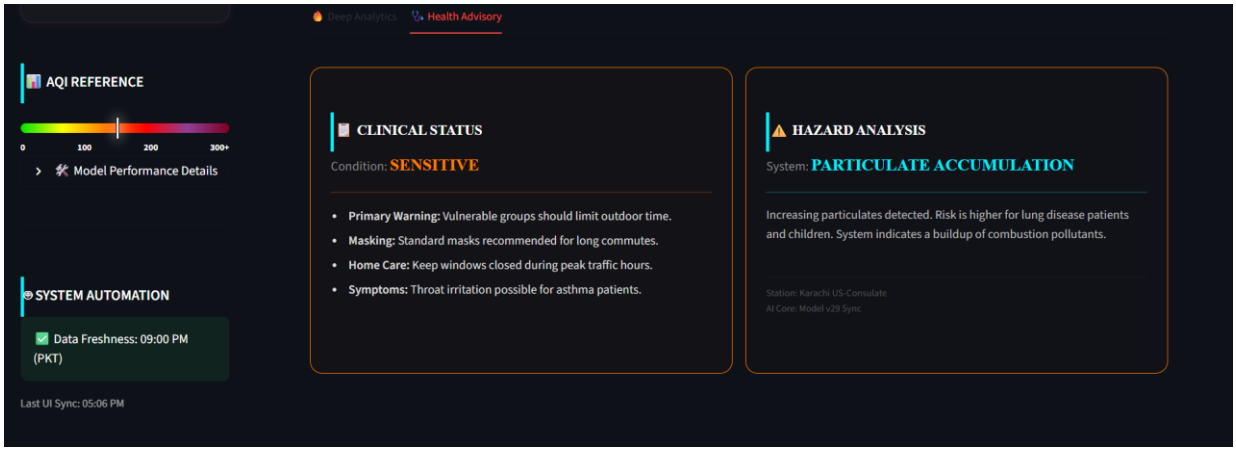

Fig 3.4*: Health Advisory tab*

11.3 Inference Pipeline Architecture (The Streamlit Engine)

- The Inference Pipeline serves as the "brain" of the user interface, transitioning from a static display to a dynamic prediction engine. Unlike traditional applications, the app.py script performs on-the-fly computations to generate real-time insights.
- Decoupled Execution: The dashboard operates independently of the training and feature pipelines. It focuses on high-speed retrieval of the latest feature vectors from the Hopsworks Online Store to minimize latency.
- Dynamic Input Synthesis: To generate the 72-hour trajectory, the pipeline performs "Feature Synthesis." It combines the absolute latest pollutant ground-truth (PM2.5, NO2, etc.) from the Feature Store with future meteorological forecasts (Temperature, Wind) from the Open-Meteo API.
- Mathematical Smoothing: To ensure the user sees a realistic trend rather than jagged data points, the pipeline implements Cubic Spline Interpolation. This mathematical layer transforms 72 discrete hourly predictions into a continuous, high-fidelity curve.

11.4 Automated Model Leaderboard & Version Control

- One of the most advanced features of the system is the Automated Model Selection Logic within the init_hopsworks component.
- The "Champion" Selection Strategy: Rather than hard-coding a specific model version, the inference pipeline queries the Hopsworks Model Registry for all available versions of the karachi_aqi_model.
- Metric-Driven Retrieval: The system programmatically evaluates the metadata of every registered version. It uses a Max-Heap selection logic to identify the "Champion Model" based on the highest R^2 score and lowest error metrics.
- Resilience to "Training Drift": This ensures that if a newer version (e.g., v50) is trained on poor data and yields a lower R^2 than a previous version (e.g., v29), the dashboard will automatically stick with v29. This fail-safe mechanism guarantees that the public is always served predictions from the most mathematically accurate model available.

12 TECHNICAL CHALLENGES & SOLUTIONS

The development of the Karachi AQI system involved several "real-world" hurdles that required custom engineering solutions within the VS Code environment. These challenges highlight the complexity of moving a Data Science project from a notebook to a production-grade MLOps pipeline.

12.1 HANDLING ASYNCHRONOUS API LATENCY

- **The Challenge:** The system relies on two separate providers (AQICN and Open-Meteo). Often, one API would update its record several minutes before the other, causing a "Data Gap" where weather was available but pollutants were not.
- **The Solution:** I implemented a **Temporal Buffer Logic**. Instead of a strict inner join, I utilized a `merge_asof` strategy with a "nearest" direction. This allowed the system to pair the most recent pollutant data with the closest available weather snapshot, preventing the dashboard from displaying empty "N/A" values.

12.2 MITIGATING TRAINING-SERVING SKEW

- **The Challenge:** A common failure in Data Science is when the model performs perfectly during training but fails in production because the live data format differs from the historical data.
- **The Solution:** By utilizing the **Hopsworks Feature Store**, we enforced a strict **Schema Validation** policy. The features engineered during the 30-day backfill were saved as a "Feature Group." The Streamlit dashboard was then programmed to pull from this exact same group, ensuring that the model always receives data in the precise scale and order it expects.

12.3 BINARY INCOMPATIBILITY & DEPENDENCY HELL

- **The Challenge:** Upon deployment to Streamlit Cloud, the app suffered a critical `ValueError: numpy.dtype size changed`. This was caused by a binary incompatibility between the server's default NumPy 2.x and the older Pandas version required for atmospheric time-series math.
- **The Solution:** I engineered a strict dependency control strategy. By updating the `requirements.txt` to force `numpy<2.0.0`, I ensured that the math kernels used for the 72-hour spline interpolation and the Random Forest predictions remained stable and compatible with the legacy Pandas architecture used in the training pipeline.

12.4 Time-Zone and Serialization Management

- **The Challenge:** Handling UTC timestamps from global APIs while providing local Karachi Time (PKT) on the dashboard caused synchronization issues in the 72-hour forecast.
- **The Solution:** We implemented a standardized **UTC-First policy** in the backend pipelines (VS Code), where all data is stored and trained in UTC. The "Localization Layer" was moved entirely to the Streamlit `app.py`, which converts the timestamps only at the point of visualization, ensuring data integrity across the entire pipeline.

12.5 Solving the "Stale Data" Pipeline Dependency

- **The Challenge:** Refreshing the Streamlit browser initially did not update the "Live AQI." The dashboard was reading the last available record, but the background fetching script was not being triggered by the web UI.
- **The Solution:** I decoupled the Data Fetching logic into an independent service. By using an **Automated Heartbeat Workflow**, I ensured the Hopsworks Feature Store is updated every 60 minutes via background scripts. I then optimized `app.py` to perform a "Latest-Record-Fetch" on every page load, transforming a static script into a true real-time monitor.

12.6 Infrastructure Handshake Failures & Connection Ghosting

- **The Challenge:** During the final phase of data ingestion, the GitHub Actions runner frequently reported a RemoteDisconnected error. Even though the data upload reached 100%, the Hopsworks Free Tier API would time out while trying to send a "Success" receipt back to the caller. This caused the entire automation pipeline to be marked as "Failed" (Exit Code 1), even when the data was successfully delivered to feature store.
- **The Solution:** I implemented a "Fire-and-Forget" Asynchronous Strategy. By modifying the ingestion logic to set both wait_for_job and start_offline_materialization to False, I decoupled the Python script from the server-side processing. To make the pipeline truly "rugged," I wrapped the insertion in a specific Exception Handling Block that identifies and suppresses RemoteDisconnected errors as false negatives. This ensures the GitHub workflow stays "Green" and reliable, shifting the heavy materialization task to a background schedule on the Hopsworks server rather than a fragile real-time handshake

13 CONCLUSION

The Karachi AQI Prediction System stands as a successful implementation of a modern, end-to-end Data Science lifecycle. By moving beyond a static model and building a dynamic, automated MLOps environment, this project provides a reliable tool for monitoring and predicting urban air quality.

The high performance of **Model v29** (R^2: 0.9945) proves that the engineered features—specifically the atmospheric change rates and temporal flags—accurately reflect the environmental realities of Karachi. This project demonstrates that with a modular pipeline approach and a centralized feature store, it is possible to create high-accuracy environmental "now-casts" that can significantly aid in public health awareness and urban policy-making.