Karachi Real-Time 3 Days Air Quality Monitoring and Forecasting System

A Machine Learning Approach using Ensemble Modeling with Automated Data Pipelines

PREPARED BY

Syeda Shinza Wasif

Data Science Intern

INSTRUCTOR

Muhammad Mobeen

ASC @10Pearls

Table of Contents

1. Introduction

The **Karachi AQI Predictor** is a sophisticated, data-driven solution designed to monitor, calibrate, and forecast air quality in Karachi, Pakistan. By integrating real-time sensor data with advanced machine learning algorithms, the project provides a comprehensive overview of current atmospheric conditions.

Unlike standard global air quality trackers, this system uses a **Dual-Check Calibration** method. It adjusts raw sensor data based on Karachi's unique coastal humidity profiles to provide an "EPA-Equivalent" Air Quality Index (AQI), ensuring that the information provided is both localized and highly accurate for the city's residents.

2. Why This Project is Needed

Karachi, one of the world's most populous megacities, faces a critical environmental challenge in the form of seasonal smog and year-round particulate matter pollution.

- **Public Health Protection:** Air pollution is a "silent killer." By providing a **72-hour forecast**, this project allows sensitive groups (children, the elderly, and those with respiratory issues) to plan their outdoor activities and minimize exposure.
- **Localized Accuracy:** Global AQI models often overlook Karachi's high coastal humidity, which can cause low-cost sensors to over-report pollution levels. This project's Dynamic Correction Logic (adjusting factors between 1.15x and 1.42x) ensures the data reflects reality on the ground.
- **Data Scarcity:** There is a lack of accessible, real-time, and historical air quality data in the public domain for Karachi. This project fills that gap by maintaining a Historical Data Log and a Model Performance Registry.
- **Predictive Insights:** Instead of just reacting to current smog, the use of XGBoost and LSTM models helps the city move toward a proactive stance, predicting "hazardous" days before they happen.

3. Tech Stack:

- Python 3.11+ (Core Programming Language)
- Streamlit (Frontend Dashboard Framework)
- MongoDB Atlas (Cloud NoSQL Database)
- TensorFlow / Keras (ANN and LSTM Deep Learning Models)
- Scikit-learn (Data Scaling, Splitting, and Multi-Output Regression)
- DagsHub (Remote Model Registry and MLflow Hosting)
- Open-Meteo APIs (Air Quality, Weather Archive, and Forecast Data)
- Plotly (Graph Objects & Express) (Interactive Data Visualization)
- Pandas & NumPy (Data Manipulation and Numerical Processing)
- GitHub (Version Control and Source Code Management)
- Streamlit Cloud (Automated Deployment and Hosting)

4. Data APIs Used

The project relies on external APIs for both historical and real-time environmental data:

- **Open-Meteo Air Quality API:** Used to fetch PM2.5 concentrations.
- **Open-Meteo Archive API:** Used for fetching historical meteorological data (temperature, relative humidity, wind speed).
- **Open-Meteo Forecast API:** Used to pull real-time 72-hour weather forecasts for predictive modeling.

5. Project Workflow

The system is orchestrated through a modular architecture consisting of six specialized pipelines, ensuring data freshness, model accuracy, and seamless deployment:

- **Backfill Pipeline:** Fetches historical data (starting from 6 months prior to 8 days ago) to initialize the MongoDB database. This serves as the foundation for training the initial baseline models and establishing historical trends.
- **Feature Pipeline:** A recurring automated process that fetches fresh weather and air quality data. It computes advanced features such as rolling averages and time-lagged variables, performing bulk upserts to MongoDB to keep the Feature Store updated.
- **Training Pipeline:** Loads the processed datasets from MongoDB to train an ensemble of models (XGBoost, ANN, and LSTM). It evaluates performance using metrics like MAE, RMSE, and R^2.
- **Inference Pipeline:** Pulls the latest live weather forecasts and recent Feature Store records.
- **CI/CD Pipeline (Automation):** Powered by GitHub Actions, this pipeline automates the entire lifecycle. It triggers the Feature Pipeline every hour and the Training Pipeline daily.
- **Web Dashboard (The Interface):** A real-time Streamlit application that serves as the user interface. It pulls the 72-hour forecasts from MongoDB, displays interactive SHAP explainability chart, and provides visual alerts for hazardous AQI levels in Karachi.

6. How AQI is Calculated

The project uses a multi-layered approach to calculate the Air Quality Index (AQI):

- **Official EPA Formula:** Raw PM2.5 concentrations are processed using the official US EPA 2024 breakpoints to generate a standard AQI.
- **Dynamic Correction (Local Calibration):** A city-specific adjustment factor (base 1.25) is applied to the raw AQI. This factor is dynamically adjusted based on Karachi's coastal humidity:
    - **High Humidity (>85%):** Factor is reduced (dampened) by 15% because moisture can cause sensors to over-read.
    - **Low Humidity (<40%):** Factor is increased by 15% to account for dust missed by sensors in dry air.

- **Smog Index:** A custom feature calculated by dividing humidity by wind speed (multiplied by a "winter" flag), as humidity traps particles while wind clears them.

7. Internship Challenges & Solutions

Developing a high-accuracy environmental prediction system in a real-world setting presented several technical and infrastructural hurdles. Below is a summary of the key problems faced and the engineering decisions made to solve them.

I. Infrastructure Pivot: From Hopsworks/Vertex AI to MongoDB

- **The Problem:** Initial plans involved using Hopsworks and Vertex AI for feature storage and model serving. However, these platforms required paid tiers for full functionality. Additionally, setting up Vertex AI through BigQuery introduced excessive complexity and cost for the project's scope.
- **The Solution:** To maintain a robust yet cost-effective pipeline, the architecture was shifted to MongoDB Atlas. MongoDB provided the necessary flexibility for a NoSQL document store, allowing for seamless storage of both real-time sensor data and complex 72-hour forecast arrays in a single collection.

II. Model Selection: XGBoost, LSTM, and ANN

- **The Problem:** Air quality data is non-linear and highly dependent on both historical trends and sudden weather changes, making simple regression models insufficient.
- **The Solution:** An ensemble approach was chosen to capture different data patterns:
    - **XGBoost:** Chosen for its superior performance with tabular data and its ability to handle "feature importance," helping identify that wind speed and humidity are the primary drivers of Karachi's smog.
    - **LSTM (Long Short-Term Memory):** Specifically selected to capture the temporal sequences and "memory" in the data, as past air quality levels (lags) are strong predictors of future levels.
    - **ANN (Artificial Neural Network):** Implemented to model complex, non-linear relationships between weather variables like the "Smog Index" and AQI results.

III. Achieving R^2 Accuracy through Fine-Tuning

- **The Problem:** Early model iterations showed low accuracy because raw PM2.5 data in Karachi is highly volatile due to coastal winds and varying humidity.
- **The Solution:** Accuracy was boosted to a 0.76 R^2 score (which is considered excellent for environmental forecasting where many external factors are unpredictable) through several steps:
    - **Feature Engineering:** Introducing the Smog Index (Humidity/Wind Speed) and Cyclic Time Features (sine/cosine of hours) provided the models with better context.
    - **Hyperparameter Tuning:** Adjusting learning rates and tree depth in XGBoost, and adding Batch Normalization and Dropout layers in the ANN to prevent overfitting.

    - **Standardization:** Using a StandardScaler to ensure that features like temperature (0-40) didn't overwhelm features like AQI change rates.

IV. Tackling Future AQI Predictions (The 72-Hour Challenge)

- **The Problem:** To predict the next 3 days of AQI, the model requires future weather data (Temperature and Wind Speed) which obviously does not exist in the historical database yet.
- **The Solution:**
    - **Live Weather Integration:** The inference script was programmed to fetch a real-time 72-hour weather forecast from the Open-Meteo API at the moment of prediction.
    - **Feature Lead Mapping:** During training, "future leads" were created by shifting historical weather data backwards, teaching the model how a temperature change *now* affects AQI *later*.
    - **Inference Loop:** The predict.py script combines the latest MongoDB actuals with these live weather forecasts to generate 72 individual hourly predictions, which are then averaged into a 3-day outlook for the dashboard.