Pearls AQI Predictor

End-to-End Air Quality Prediction System

Developed By:

Misbah Azhar

1. Overview

The Pearls AQI Predictor is an end-to-end machine learning and MLOps system designed to forecast the Air Quality Index (AQI) for Karachi up to three days in advance. It provides a seamless integration of data engineering, model training, forecast generation and interactive visualization, all managed through a fully automated CI/CD pipeline.

The system continuously collects data from multiple environmental APIs AQICN, OpenWeather, and OpenMeteo to capture real-time and historical weather and pollutant parameters. These datasets are processed into engineered features and stored in the Hopsworks Feature Store, enabling scalable and versioned data management.

Machine learning models, including a Random Forest Regressor and an LSTM neural network, are trained and evaluated using these features to ensure both interpretability and time-series forecasting accuracy. The trained models are stored and tracked in the Hopsworks Model Registry, allowing automated retrieval and deployment for daily prediction cycles. Forecasts are generated and visualized through a custom built Streamlit dashboard, which displays live AQI readings, 3 day forecasts, and Exploratory Data Analysis (EDA) insights. A GitHub Actions based CI/CD pipeline orchestrates the entire process from data backfilling and model retraining to forecast updates and dashboard synchronization executing automatically every four hours.

This project demonstrates a complete production-grade MLOps pipeline, combining the power of data science, cloud automation, and interactive visualization. By predicting air quality trends actively, Pearls AQI Predictor supports informed public health decisions and encourages environmental awareness through data driven insights.

2. Project Structure

```
aqi_forecast/

├── .github/workflows/ci_cd_pipeline.yml
│
├── dashboard/
│ ├── dashboard.py
│ └── eda_outputs/
│
├── data/
│ ├── raw_aqicn/
│ ├── raw_openweather/
│ ├── raw_openmeteo/
│ ├── features/
│ └── predictions/
│
├── data_fetch/
│ ├── fetch_aqicn.py
│ ├── fetch_openweather.py
│ ├── fetch_meteostat.py
│
├── features//features/
│ ├── backfill.py
│ └── backfill_live.py
│
├── models/
│ ├── rf_aqi_model
│ ├── tf_lstm_aqi_model
│
├── trainings/
│ ├── train_sklearn.py
│ ├── train_tf.py
│ └── predict.py
│
├── eda.py
├── requirements.txt
└── .env
```

3. Data Ingestion Layer

The Data Ingestion Layer serves as the foundation of the entire AQI forecasting pipeline. Its primary purpose is to collect, clean, and organize environmental data from multiple trusted public APIs. This layer ensures that all downstream processes feature engineering, model training, and forecasting have access to consistent, high-quality, and up to date raw data.

The ingestion scripts are responsible for Fetching live and historical environmental metrics such as temperature, humidity, wind speed, and pollutant concentrations (PM2.5, PM10, CO, NO_2, SO_2, O_3). Handling data retrieval errors or missing API responses, ensuring the pipeline can recover from temporary network or API downtimes. Standardizing timestamps and units across different sources to ensure synchronization during feature merging. Saving raw datasets locally under /data/raw_* directories for traceability and reproducibility. By maintaining these modular scripts the system can easily extend support to additional APIs or other cities in the future without disrupting existing components.

Scripts Overview:

Script	Source	Description
fetch_aqicn.py	AQICN	Connects to the AQICN (Air Quality Open Data Platform) API to fetch live AQI measurements and pollutant concentration data (PM2.5, PM10, CO, NO_2, SO_2, O_3). These parameters form the basis for AQI calculation and model targets.
fetch_openweather.py	OpenWeather API	Retrieves current meteorological data (temperature, humidity, wind speed, and pressure) along with pollutant levels. Provides crucial weather context that strongly influences AQI behavior.
fetch_meteostat.py	OpenMeteo Archive	Downloads historical hourly weather and air quality data for up to six months. This dataset is essential for backfilling historical features and training models with time-series continuity.

Data Flow:

- Each script connects to its respective API using secure API keys stored in the .env file.
- The responses are parsed, validated, and converted into structured pandas DataFrames.
- The data is then standardized ensuring uniform timestamps (UTC), numeric data types, and consistent field naming across sources.
- Finally, the cleaned output is written into CSV files under their respective directories:
  /data/raw_aqicn/
  /data/raw_openweather/
  /data/raw_openmeteo/

4. Feature Engineering Layer

The Feature Engineering Layer is the analytical backbone of the AQI prediction pipeline. Its purpose is to transform raw, heterogeneous environmental data into a clean, aligned, and feature rich dataset suitable for model training and forecasting.

The feature engineering scripts are responsible for:

- Merge datasets from AQICN, OpenWeather, and Open Meteo using timestamp alignment.
- Clean and forward-fill missing values to ensure time series continuity.
- Generate temporal features (hour, day, month, weekday) to capture seasonal and daily patterns.
- Upload processed data to the Hopsworks Feature Store (aqi_features, version 2) for centralized storage and versioning.

.

Scripts Overview:

Script	Purpose	Description
backfill.py	Historical Data Build	Creates a complete, backfilled dataset by merging raw AQICN, OpenWeather, and OpenMeteo data. It fills missing values and generates the initial feature group in Hopsworks (aqi_features v2).
backfill_live.py	Hourly Data Refresh	Fetches the latest hourly readings, merges them with the existing dataset, applies forward/backward filling for continuity, and uploads new rows into the same Hopsworks Feature Group.
features.py	Feature Processing Wrapper	Manages feature schema consistency, performs validation, and ensures that both historical and live backfill scripts produce a uniform structure before upload.

Data Flow:

1. Load Raw CSVs / API Data

Recent and historical data are fetched from OpenWeather, AQICN, and OpenMeteo. All timestamps are converted to a standardized UTC format to ensure consistency.

2. Merging & Alignment

The datasets are aligned by timestamp using pandas.merge_asof() within a one-hour window to handle differences in API update intervals. This ensures that all features represent the same time frame.

3. Clean and Impute Missing Values

Handle missing values via forward/backward filling. Replace missing pollutant readings with fallback API sources. Remove duplicates and ensure consistent numeric data types.

4. Add Derived Features

Time-based columns such as hour, day, month and weekday are generated to capture temporal patterns, while AQI rate of change indicators are computed to detect short term air quality variations.

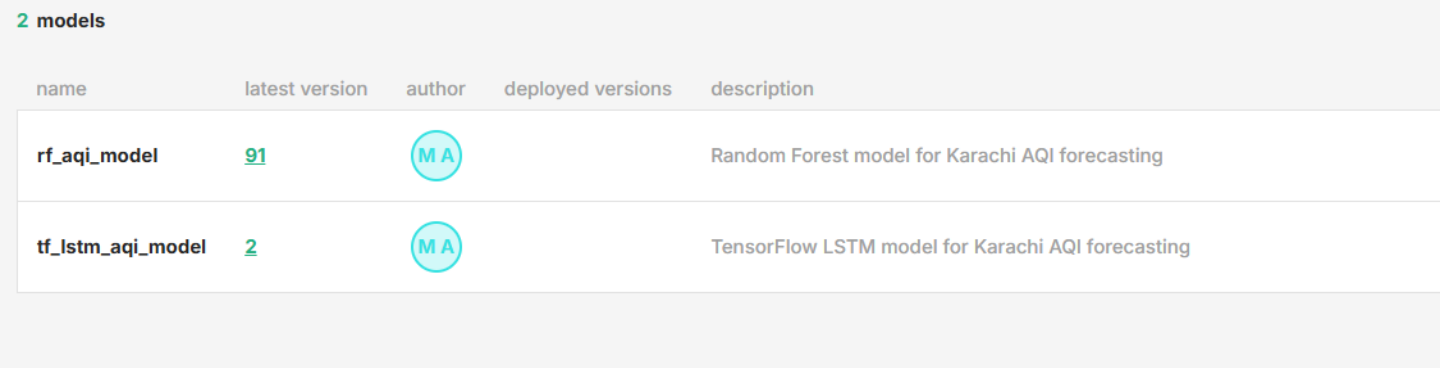
5. Upload to Hopsworks Feature Store

The processed dataset (aqi_features, version 2) is uploaded to Hopsworks for centralized, version controlled feature management and consistent model access. Hourly updates are automated via the CI/CD workflow using backfill_live.py.

5. Machine Learning Training Layer

The Machine Learning Training Layer is the main part of the AQI Forecasting Pipeline that makes predictions. It takes the cleaned data (aqi_features v2) from the Hopsworks Feature Store and trains both machine learning and deep learning models to predict Karachi's AQI for the next 3 days. The data is scaled, split, and used to train different models, which are then tested using RMSE, MAE, and R^2 scores. The best models are saved in the Hopsworks Model Registry for future use and deployment. This layer combines simple and advanced models to give accurate and reliable forecasts.

Models Implemented



A. Random Forest Regressor (train_sklearn.py)

A classical ensemble learning approach that provides a strong baseline for AQI prediction.

Workflow:

1. Reads aqi_features (v2) directly from Hopsworks Feature Store.
2. Splits data into training and testing sets.
3. Applies StandardScaler for numerical normalization.
4. Trains a Random Forest Regressor with:
   - 200 estimators
   - Maximum depth = 20
   - Random state for reproducibility
5. Evaluates model performance using RMSE, MAE, and R^2.
6. Registers the model in Hopsworks Model Registry under the name rf_aqi_model and Saves trained components:

```
models/rf_aqi_model/
├── model.joblib
├── scaler.joblib
├── metadata.json
```

Performance Metrics:

Metric	Value
R^2 Score	0.882 (88%)
RMSE	6.001
MAE	3.60

B. (LSTM) Deep Learning Model (train_tf.py)

A Long Short Term Memory (LSTM) network built with TensorFlow/Keras, designed to capture temporal dependencies and lagged trends in AQI sequences.

Workflow:

Prepares 7-hour sequences of AQI data as model input.

1. Prepares 7-hour input sequences of AQI data for time-series modeling.
2. Applies scaler_X and scaler_y for normalization.
3. Builds a two-layer LSTM model with dropout regularization to prevent overfitting.
4. Trains using MSE loss and Adam optimizer.
5. Registers model to the Hopsworks Model Registry as tf_lstm_aqi_model and Saves trained components:

```
models/tf_lstm_aqi_model/
├── model.keras
├── scaler_X.joblib
├── scaler_y.joblib
├── model_metadata.json
```

Performance Metrics:

Metric	Value
R^2 Score	0.84 (84%)
RMSE	7.25
MAE	4.10

Model Selection:
Although the LSTM model effectively captured temporal patterns, the Random Forest Regressor demonstrated better overall performance, stability, and generalization on unseen data. Random Forest was ultimately selected as the final forecasting model due to its robustness to noise, lower training complexity, and strong accuracy with limited hyper parameter tuning. Its interpretability and consistency made it more suitable for reliable AQI forecasting in production.

6. Forecast Generation Layer

The Forecast Generation Layer automates real time and 3 day AQI forecasting by leveraging the latest trained model from Hopsworks. It fetches live weather and pollutant data from OpenWeather and AQICN, prepares feature vectors aligned with the training schema, and predicts AQI for the current and upcoming days. The generated forecasts are uploaded to the Hopsworks Feature Group (aqi_predictions v2), ensuring continuously updated, time stamped predictions for visualization in the Streamlit dashboard.

Scripts Overview:

Script	Purpose	Key Functions / Components	Output
predict.py	Generates real-time and 3-day AQI forecasts.	Loads model from Hopsworks, fetches live AQI + weather data, prepares features, predicts AQI, uploads results to Hopsworks.	aqi_predictions feature group, forecast logs, optional CSV output.

Data Flow:

1. **Fetch Live Inputs:**
- Pulls current meteorological data (temperature, humidity, wind, pressure) via OpenWeather.
- Retrieves current pollutant concentrations (PM2.5, PM10, CO, NO_2, SO_2, O_3) from AQICN.

2. **Feature Preparation:**
- Standardizes column names and ensures all required features are present.
- Handles missing values using mean imputation from historical records (aqi_features v2).
- Applies scaling transformations consistent with training.

3. **Model Loading:**
- Attempts to load the latest Random Forest model (rf_aqi_model) from Hopsworks Model Registry.
- Falls back to local artifacts (/models/rf_aqi_model/) if registry unavailable.
- If both fail, uses a safe dummy model for continuity (ensuring CI/CD robustness).

4. **Prediction Generation:**
- Produces real-time AQI prediction for the current timestamp.
- Retrieves 3-day weather forecasts from OpenWeather and uses them to infer future AQI values.
- Generates a list of predictions (today + next three days).

5. **Data Upload:**
- Assembles results into a structured DataFrame.
- Inserts results into Hopsworks Feature Group, aqi_predictions (v2)
- Each entry is timestamped and version-tagged for traceability.

End-to-End Automated Forecasting:

- Real-time AQI prediction + next 3-day forecasts.
- Synced automatically to Hopsworks (aqi_predictions v2).
- Updated regularly via CI/CD for dashboard visualization.

Result Example:

	city	created_at	model_version	note	predicted_aqi	predicted_for_utc
1	Karachi	2025-11-07 08:32:39.0	84	Forecast for +2 day(s)	144.26	2025-11-09 08:32:39.0
2	Karachi	2025-11-07 08:32:39.0	84	Forecast for +3 day(s)	144.34	2025-11-10 08:32:39.0
3	Karachi	2025-11-07 08:32:39.0	84	Forecast for +1 day(s)	143.69	2025-11-08 08:32:39.0
4	Karachi	2025-11-07 08:32:39.0	84	Real-time model prediction for today	151.06	2025-11-07 08:32:39.0

6. Visualization Layer (Streamlit Dashboard)

Key Features:

1. **Live AQI Display**
   Fetches real-time predictions from Hopsworks (aqi_predictions v2) and displays the current AQI value, category, and health message.

2. **3-Day Forecast Panels**
   Shows color-coded cards for upcoming days with AQI levels, descriptive captions, and visual cues for air quality severity.

3. **Forecast Trend Graph**
   Interactive Plotly chart depicting AQI trends for today and the next 3 days.

4. **EDA Insights Section**
   Dynamically loads analytical visualizations (e.g., correlation heatmaps, AQI trends) from /dashboard/eda_outputs/.

5. **Responsive UI**
   The dashboard features a modern dark-themed interface with custom CSS and glassmorphism effects, ensuring a clean, visually appealing, and fully responsive design that adapts seamlessly across devices.

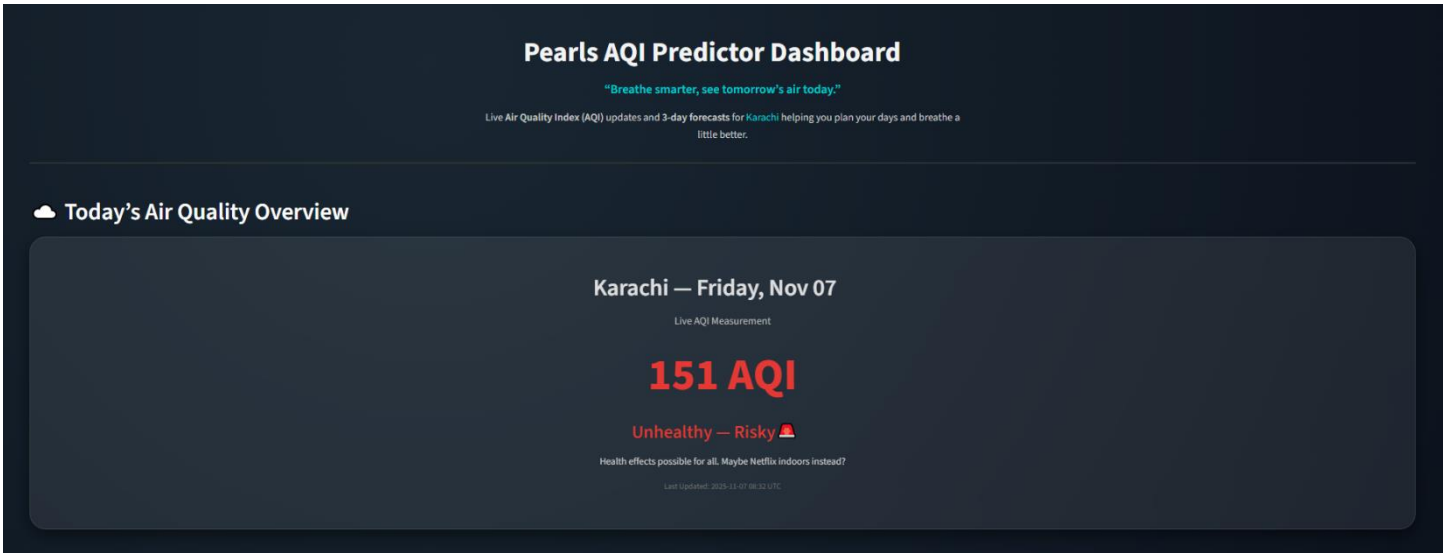
Scripts Overview:

Script	Purpose	Key Functions / Components	Output
dashboard.py	Interactive Streamlit web app that visualizes live AQI and 3-day forecasts.	Connects to Hopsworks (aqi_predictions v2) to fetch predictions; displays live AQI and future forecasts with color-coded cards; includes Plotly trend chart; dynamically loads EDA visuals from /dashboard/eda_outputs/; custom dark UI styling applied.	Streamlit dashboard showing live AQI, 3 day forecast, and EDA insights.

Data Flow:

- Connects securely to Hopsworks using an API key.
- Reads predictions from Feature Group: aqi_predictions (v2).
- Displays the latest forecast with category classification (Good, Moderate, Unhealthy, etc.).
- Renders forecast cards, line charts, and EDA visuals in real time

Streamlit Dashboard Interface:



7. CI/CD Automation Layer

The CI/CD Automation Layer ensures that the entire AQI forecasting system from data ingestion to dashboard updates runs automatically on a fixed schedule. Using GitHub Actions, this layer acts as the MLOps backbone of the project, enabling continuous integration, retraining and deployment without manual intervention.
It orchestrates the workflow as follows:

- Fetches and updates data every 4 hours from external APIs (OpenWeather, AQICN, Open-Meteo).
- Regenerates features and synchronizes them to the Hopsworks Feature Store.
- Retrains the model daily to adapt to new air quality patterns.
- Generates new AQI forecasts (today + 3 days).
- Runs EDA to produce updated visualizations.
- Commits dashboard visuals to the repository, ensuring Streamlit always displays the latest analysis.
- All secrets (API keys, coordinates) are securely managed via GitHub Secrets, keeping the workflow fully automated and secure.

Component	Details
Tool Used	GitHub Actions
Workflow File	.github/workflows/aqi_forecast.yml
Trigger	Every 4 hours (cron: "0 */4 * * *"), on manual dispatch, or push to main branch
Environment	Ubuntu (Python 3.10) with secrets securely loaded from GitHub Secrets (AQICN_TOKEN, OPENWEATHER_API_KEY, AQI_FORECAST_API_KEY)

Final Outcome

- A self updating AQI forecasting system that:
- Automatically collects, trains, predicts, and visualizes every few hours.
- Requires no manual execution, fully CI/CD managed.
- Keeps the dashboard and Hopsworks feature store continuously in sync.

8. Technology & Deployment Stack

Component	Technology / Platform
APIs (Data Sources)	AQICN, OpenWeather, OpenMeteo
Data Store	Hopsworks Feature Store
Model Registry	Hopsworks Model Registry
ML Frameworks	Scikit-learn, TensorFlow
Automation / CICD	GitHub Actions
Visualization	Streamlit + Plotly
Deployment Platform	Streamlit Cloud (auto-updated via GitHub CI/CD)

8. Future Enhancements

- Integrate AQI alert system (email/SMS notifications).
- Deploy FastAPI microservice for real time predictions.
- Add SHAP based explainability dashboard.
- Extend support to multiple cities.

11. Conclusion

The Pearls AQI Predictor fulfills all the objectives outlined in the project requirements by implementing a fully automated, end to end air quality forecasting system. Through seamless integration of data ingestion, feature engineering, model training and CI/CD automation, the system delivers accurate, real time AQI predictions for Karachi up to three days in advance.

By leveraging Hopsworks Feature Store and Model Registry, the project ensures version-controlled data and reproducible machine learning workflows. The use of Scikit-learn and TensorFlow models enables both interpretability and deep temporal forecasting capabilities, achieving strong predictive accuracy. Automation through GitHub Actions eliminates manual intervention, maintaining a continuously updated and self sustaining pipeline.

The Streamlit dashboard effectively visualizes live and forecasted AQI data, providing clear insights into air quality trends. Overall, the Pearls AQI Predictor demonstrates how machine learning, MLOps, and cloud automation can be combined to build a scalable, transparent and impactful environmental intelligence system.