Karachi AQI Prediction System

Version: 1.0
Date: 15 Aug 2025
Author: Sameer Kamani

\

Table of Contents

Overview

This project is a complete air quality prediction system that automatically:

- Collects weather and pollution data
- Processes it into features
- Trains multiple machine learning models
- Provides predictions through a web interface

The system is designed to run continuously with **hourly automated updates** and **daily model retraining**.

Project Structure

1. Data Collection (`Data_Collection/`)

- **data_fetch.py** → Main script (runs every hour)
- Connects to **OpenMeteo API** to fetch:
  - Air quality data (PM2.5, PM10, NO_2, SO_2, CO, O_3)
  - Weather data (temperature, humidity, precipitation)
- Data is processed into **daily averages**
- Stored in **CSV** (readable) and **Parquet** (efficient)
- Fetches **only new data**, preserving history

---

2. Feature Store (`feature_repo/`)

- Uses **Feast (Feature Store system)**
- **karachi_features_new.py** defines dataset structure

- Creates engineered features:
  - Daily averages
  - Weather conditions
  - Derived/calculated values
- Provides easy and organized data access

---

3. Data Processing (`Data/`)

- **feature_store/** → Processed data files
- **processed/** → Intermediate data
- **raw/** → Original collected data

---

4. Machine Learning Models (`Models/`)

- **train_lightgbm.py** → LightGBM models (predict AQI 1–3 days ahead)
- **train_hgbr.py** → Histogram Gradient Boosting models
- **train_linear.py** → Linear regression models
- **train_rf.py** → Random Forest models
- **stacking_linear_lightgbm.py** → Ensemble model creation
- **predict_realtime.py** → Real-time predictions

---

5. Model Registry (`Models/Models/registry/`)

- Stores trained models, versions, metrics, predictions
- Separate files for each model type (LightGBM, HGBR, Linear, RF)
- Ensemble weights are tracked here

---

6. Exploratory Data Analysis (`Models/EDA/`)

- **run_eda.py** → Performance and data analysis
- Generates:
  - Visualizations & reports
  - Feature importance rankings
  - Accuracy summaries per model

---

7. Web Application (`WebApp/`)

- **Backend** (`WebApp/Backend/app`)
  - **main.py** → Core server
  - **model_loader.py** → Loads trained models
  - **feast_client.py** → Connects to feature store
- **Frontend** (`WebApp/Frontend`)
  - **gradio_app.py** → Web interface (predictions, trends, graphs)

---

8. Automation (GitHub Workflows)

- **features-hourly.yml** → Collects new data every hour
- **train-daily.yml** → Retrains models daily (2:15 AM)

System Workflow

1. **Data Collection (Hourly)**
   - Fetches AQI + weather data
   - Converts into daily features
   - Saves CSV + Parquet
   - Updates Feast feature store
2. **Feature Engineering**
   - Adds:
     - Time-based (month, season, weekday)
     - Lag features (previous AQI)
     - Rolling averages (3-day)
     - Log transformations
   - Improves pattern recognition
3. **Model Training (Daily)**
   - Trains **LightGBM, HGBR, Linear, RF**
   - Last **90 days** reserved for testing
   - Predicts **1, 2, 3 days ahead**
   - Metrics: **RMSE, MAE, R^2**

4. **Ensemble Creation**
    - Combines model predictions using **optimized weights**
    - Improves robustness and accuracy
5. **Prediction Generation**
    - Real-time predictions for **next 3 days**
    - Includes **confidence + model contributions**
6. **Web Interface**
    - Displays:
        - Current AQI
        - Historical trends
        - Future predictions
    - Allows **manual updates**

My Journey

I started this project blindly, with prior experience only in **classification**, not regression. Initially:

- Extracted **4.5 years of data** using **OpenWeather API**
- Created **150+ features** (too many, caused poor performance)
- Tried multiple models (**SARIMA, LSTM, RNN, GRU, XGBoost, CatBoost**) → all gave **$R^2 < 0.4$**

I later discovered through **SHAP analysis** that many features were **hurting performance**. After:

- Switching fully to **OpenMeteo API** (cleaner data, includes weather)

- Reducing to **33 key features**
- Building an **ensemble model**

→ My accuracy improved significantly.
I'm especially proud of the ensemble, which gives robust predictions.

Technical Components

- **Data Sources**: OpenMeteo API, CSV/Parquet storage, Feast Feature Store
- **ML Frameworks**: LightGBM, Scikit-learn, Custom ensembles
- **Web Tech**: FastAPI (backend), Gradio (frontend), RESTful API
- **Automation**: GitHub Actions, Python pipelines, Feast
- **Data Processing**: Pandas, NumPy, custom pipelines

Model Performance (Aug 19, 2025)

1. Ensemble Model (Best Overall)

Horizon	RMSE	MAE	R^2
Day 1	4.03	3.18	0.88
Day 2	9.32	7.24	0.32
Day 3	10.67	8.78	0.07

---

2. Individual Models

LightGBM

Day	RMSE	MAE	R^2

1	4.25	3.14	0.86
2	9.98	7.80	0.23
3	12.74	9.50	-0.27

Random Forest

Day	RMSE	MAE	R^2
1	4.12	3.26	0.87
2	9.36	7.28	0.32
3	11.98	8.92	-0.13

Histogram Gradient Boosting (HGBR)

Day	RMSE	MAE	R^2
1	4.09	3.15	0.87
2	9.90	7.48	0.24
3	10.67	8.78	0.07

Linear Regression

Day	RMSE	MAE	R^2
1	4.99	4.01	0.81
2	9.85	7.78	0.25
3	12.33	9.82	-0.19

---

3. Ensemble Model Weights (Least Squares Optimization)

Horizon	Random Forest	HGBR	Linear	LightGBM
Day 1	**39.8%**	32.0%	24.3%	3.9%

Day 2	**82.6%**	7.7%	9.7%	0%
Day 3	0%	**70.9%**	4.7%	24.4%