Karachi AQI Prediction – 10Pearls Internship Project

Prepared by: *Muhammad Waqi Mallick*

1. Project Overview

The Karachi AQI Prediction Project was developed during my internship at 10Pearls to forecast the Air Quality Index (AQI) of Karachi using historical weather and pollutant data.

The objective of this project is to understand the environmental patterns influencing air quality and build a predictive model that can estimate AQI trends for the coming days. The solution integrates data collection, preprocessing, exploratory analysis, feature engineering, model training, and automated daily updates using GitHub Actions.

2. Data Collection

2.1 Data Source

All the data for this project—both **weather** and **pollutant**—was fetched from the **Open-Meteo API**. This API provided comprehensive daily data for variables such as:

- **Weather metrics:** temperature, humidity, wind speed, and pressure
- **Pollutants:** PM2.5, PM10, NO_2, CO, SO_2, and O_3

To ensure sufficient sample size and model stability, I fetched one year's worth of historical data for Karachi.

2.2 Feature Store

After fetching the data from the Open-Meteo API, it undergoes preprocessing — including timestamp alignment, removal of any redundant or null entries, and extraction of temporal features such as day, month, and hour to enhance the model's learning capabilities.

The processed data is then stored in MongoDB, which serves as a feature store for this project. MongoDB was chosen because of its flexibility in handling semi-structured time-series data and its high read/write performance, which makes it efficient for frequent updates and retrievals.

Initially, the project aimed to use Hopsworks as a dedicated feature store; however, due to recurring reliability issues and service downtime, MongoDB was selected as a more stable and lightweight alternative. Additionally, MongoDB's native JSON-like structure makes it easier to integrate seamlessly with Python-based data pipelines, ensuring smooth data access during both training and prediction stages.

3. Data Preparation and Exploration

3.1 Exploratory Data Analysis (EDA)

Once collected, the data was explored using tools such as pandas and matplotlib:

- df.head() and df.info() were used to verify column structure and data types.
- df.describe() helped confirm valid ranges for numeric features and detect outliers.

Since no missing or invalid entries were found, minimal cleaning was required.

Correlation heatmap: Revealed strong positive correlations between AQI and PM2.5/PM10 levels.

3.2 Feature Selection

Features were selected through **correlation analysis and domain research** rather than algorithmic selection.
I utilized the heatmap results and did some research on some already made AQI models to identify the features, in some papers they were utilizing only the pollutant features and in some they were using the weather conditions so I decided to go with a combined approach to see how it influenced the model. So the selected features were('time', 'carbon_monoxide', 'day', 'dayofweek', 'hour', 'month', 'nitrogen_dioxide', 'ozone','pm10','pm2_5','relative_humidity_2m','sulphur_dioxide', 'temperature_2m', 'us_aqi', 'wind_speed_10m'). I know there could be more filtering with the features but according to the heatmap almost every feature was having some kind of +ve or -ve influence on the aqi value.

4. Model Training and Evaluation

4.1 Models Used

Three regression models were implemented and compared:

1. **Linear Regression**
2. **Gradient Boosting Regressor**
3. **Random Forest Regressor**

The dataset was divided into **80% training** and **20% testing** sets to evaluate model generalization.

4.2 Training and Metrics

Each model was trained on the selected features, and performance was evaluated using:

- **R^2 Score (Accuracy)**
- **Mean Absolute Error (MAE)**
- **Root Mean Squared Error (RMSE)**

4.3 Results

Among the tested algorithms:

- **Random Forest Regressor** achieved the **best performance** with an **accuracy of 82.2%**, making it the final chosen model.
- Analysis of feature importance confirmed **PM2.5** as the **most dominant predictor** of AQI, followed by PM10 and humidity.

These findings align with environmental studies, validating the relationship between particulate matter and air quality degradation.

5. Web Application

A simple Flask web application was built to present the model's output and results in an accessible format. The app is designed using HTML, CSS, and Flask for backend integration.

It consists of **three main pages**:

1. **Home Page** – Provides an introduction to the project and its objectives.
2. **Predict Page** – Displays the forecasted AQI values for the next 3 days, based on the latest trained model.
3. **Results Page** – Summarizes the performance of all three trained models, showing metrics and visualizations of feature importance.

This modular design allows for easy future expansion, including direct integration with the automated data pipeline and live deployment.

6. Automation/Github Actions:

The data collection pipeline is fully automated using GitHub Actions:

- The data collection script runs daily at 9:00 PM Pakistan time, fetching the entire day's readings.
- The script stores the latest weather and pollutant data after preprocessing in MongoDB(utilizing it as a feature store).

To reduce the number of API calls and ensure data reliability, the system fetches data once per day instead of hourly. This approach avoids redundant requests since the Open-Meteo API does not update readings on an hourly basis.

A model training script is also automated to run daily at 10:00 PM, which retrains the model with the newly fetched data. This ensures that the predictive system always remains up-to-date and ready for integration with future applications.

7. Key Takeaways

- Learned to integrate real-world APIs (Open-Meteo) for environmental data acquisition.
- Understood how to design end-to-end data pipelines and automate them using GitHub Actions.
- Practiced feature analysis and model evaluation for real-life regression problems.
- Developed a functional Flask-based AQI forecasting app with automated data updates.
- Gained hands-on experience in data science project structuring, automation, and reproducibility.