Name: Humaila farooq

Air Quality Forecasting System – Project Report

1. Project Overview

This system predicts the **Air Quality Index (AQI)** for **Karachi's 10 Pearl regions** using a complete **MLOps pipeline** built with **Python**, **Hopsworks Feature Store**, and **GitHub Actions**. It automatically fetches hourly data, preprocesses it, trains and updates ML models, and displays live forecasts on a **Streamlit dashboard**.

The pipeline uses **one year of continuous OpenWeather data**, enabling **daily retraining** and **hourly forecasting** for the next 72 hours.

2. Data and Preprocessing

a) Data Sources

- **Source:** OpenWeather API (hourly data for 12 months).
- **Raw features (10):** temperature, humidity, wind speed, pressure, PM2.5, PM10, NO_2, O_3, CO, visibility.
- **Timestamps added:** hour, day, month, year, weekday.

Since OpenWeather doesn't provide U.S. EPA AQI, the **standard AQI** was computed manually using:

[

(I_{hi}-I_{lo})/(C_{hi}-C_{lo}) * (C - C_{lo}) + I_{lo}

]

where *C* is pollutant concentration.

b) Feature Engineering

From 10 raw variables, **77 total features** were generated and stored in Hopsworks:

Category	Example	Count
Raw	temperature, humidity, PM2.5	10
Time-based	hour, weekday, hour_sin, month_cos	10
Lag	standard_aqi_lag_1, lag_6, lag_72	20
Rolling stats	roll_mean_6h, roll_std_24h	15
Change rates	AQI change/pct_change	10
Interaction	temp×humidity, pm_ratio, oxidant_sum	5
Targets	next_1h, next_24h, next_72h	3

This comprehensive feature set captures both short-term and seasonal AQI dynamics.

c) Missing Data Handling

A robust imputation sequence was applied:

1. Forward-fill → backward-fill for short gaps.
2. Median fill for numeric columns.
3. "Unknown" for categorical data.

To prevent leakage, columns like `standard_aqi_next_1h`, `next_16h`, `next_72h` were dropped; the model was trained only on `standard_aqi_next_24h`.

3. Historical Backfill and Feature Store

The **backfill pipeline** reconstructs one year of history:

- Fetches hourly weather/pollution data from OpenWeather.
- Computes AQI and engineered features.
- Applies preprocessing (drops leakage, fills missing values).
- Saves output to `features_preprocessed.csv`.

- Uploads to **Hopsworks Feature Store**, each record representing one hour of data.

This ensures consistent and reproducible data for both training and live prediction.

4. Model Training and Evaluation

Models were trained on one year of data from Hopsworks and compared by accuracy:

80/20 train-test split

Model	Train R^2	Test R^2	MAE	RMSE
Random Forest	**0.9320**	**0.7000**	**17.63**	**21.76**
XGBoost	0.9959	0.5695	21.26	26.07
LightGBM	0.9884	0.5938	21.01	25.32
SARIMAX	0.9594	−3.067	57.68	80.13

Insights:

- Random Forest generalized best and became the deployed model.
- XGBoost and LightGBM slightly overfit on long-term data.
- SARIMAX performed well on 6-month data but broke down on 1-year scale.

The final model was fine-tuned for the **next-24h AQI target**, using hyperparameter tuning and feature medians. All trained artifacts are stored in **Hopsworks Model Registry**.

5. Prediction Pipeline (Recursive Forecasting)

The **predict.py** script forecasts AQI for the next **72 hours** using a **recursive approach**:

1. Load model, scaler, medians, and feature list from Hopsworks or local backup.
2. Retrieve last 48 hours of features.
3. Create 72 future timestamps.
4. Predict each hour iteratively:

```
for i in range(72):
    sub = combined.iloc[:hist_rows + i + 1]
```

```
eng = engineer_features(sub, return_last_row=True)
pred = model.predict(scaler.transform(X_row))
combined.at[hist_rows + i, "standard_aqi"] = pred
```

5. Save results to `predictions.csv`.

This **recursive prediction** reuses each predicted AQI as input for the next step. It performs well up to 24h and moderately up to 72h, where minor overfitting and drift occur.

Accuracy was highest for **6h and 24h forecasts**, confirming the model's short-term reliability.

6. Automation and CI/CD

Two GitHub Actions workflows automate the system:

- **Hourly Ingestion (`hourly_ingest_pip.yml`)** → fetches latest data, computes features, updates Feature Store.
- **Daily Training (`train_daily.yml`)** → retrains and evaluates models at 02:00 UTC, uploading logs as artifacts.

Each workflow:

- Sets up Python 3.12.7
- Installs dependencies via `requirements.txt`
- Loads API secrets (.env)
- Logs execution to GitHub Artifacts

This ensures continuous model refresh and uninterrupted predictions.

7. Dashboard and Visualization

The **Streamlit dashboard** loads `predictions.csv` to visualize forecasts:

- Displays next 72 h AQI values with color-coded categories (1 = Good to 5 = Hazardous).
- Uses **Matplotlib** and **Seaborn** for trend charts.
- Updates automatically after every CI/CD run.
- Issues warnings for **AQI > 200** (hazardous).

This real-time visualization supports public awareness and rapid response.

8. Technologies Used

Languages & Libraries: Python, Pandas, NumPy, Scikit-learn, Matplotlib, Seaborn

Models: Random Forest, LightGBM, XGBoost, SARIMAX

MLOps: Hopsworks Feature Store & Model Registry

Automation: GitHub Actions (hourly + daily)

Dashboard: Streamlit, FastAPI

Utilities: Joblib, Dotenv, Requests, Boto3

9. Summary

This project establishes a **fully automated AQI forecasting system** built on:

- One year of OpenWeather data (77 engineered features)
- Random Forest as best model for next-24 h prediction
- Recursive approach for 72-hour forecasting
- Hopsworks for centralized feature and model management
- GitHub Actions for end-to-end automation
- Streamlit dashboard for live visualization