Project Final Report

Project Title: PEARLS AQI PREDICTOR

SUBMITTED BY: Sadia Ali

Email: sadia.ali2319@gmail.com

Live Application: https://projcet-mot8ocxkoihxmufu4xnbhn.streamlit.app/

Internship Mentor: Muhammad Mobeen

Table of Contents

1.Acknowledgements

I want to thank my mentor Muhammad Mobeen at 10Pearls for their guidance throughout this project. Special thanks to the Open-Meteo API team for providing free air quality data and to the Streamlit community for making dashboard creation accessible. This project wouldn't be possible without these open-source tools and the MLOps principles that guided its development.

2.Abstract

This project builds a complete Air Quality Index (AQI) prediction system for Karachi with also showing different region of Karachi that forecasts pollution levels for the next 3 days (72 hours) through online app. The system automatically collects air quality data every hour, storing data as a feature store and model registry into MongoDB Atlas and processes it into useful features, trains 4 different types of machine learning models daily, and displays predictions through an interactive dashboard with different kinds of visualization. Everything runs automatically using GitHub Actions, making it a true end-to-end MLOps solution.

Key Achievement: Created a working system that runs 24/7 without manual intervention, predicting AQI by daily which can be access everywhere by anyone and the main thing it is made by using all free features which really makes a cost free system.

3.Background

Air pollution in Karachi has become a serious health concern especially for older and lungs patients, with PM2.5 levels often exceeding safe limits. Traditional AQI monitoring only shows current conditions, but people need **future forecasts** to plan outdoor activities. This project addresses that gap by combining:

1. **Real-time data collection** from APIs
2. **Machine learning** to predict future pollution for next three days
3. **Automated pipelines** that run without human help, every hour and every day.
4. **User-friendly dashboard** for citizens for better visualization and user friendly.

4.Problem Statement

The main problem is:

How can we automatically predict the next 72 hours of AQI using real-time air pollution data, without manual intervention?

Problems Faced

- Collecting air quality data continuously was difficult.
- Raw data was not clean and could not be used directly.
- Features needed to be generated automatically.
- The model was not updating when new data arrived.
- GitHub Actions pipeline was not running properly at first.
- The app was sometimes using **old models and old data**.
- Connecting the automated pipeline with the Streamlit app was challenging.
- Database keys and API keys caused errors during deployment.
- Showing updated results clearly on the dashboard was difficult.

Solutions

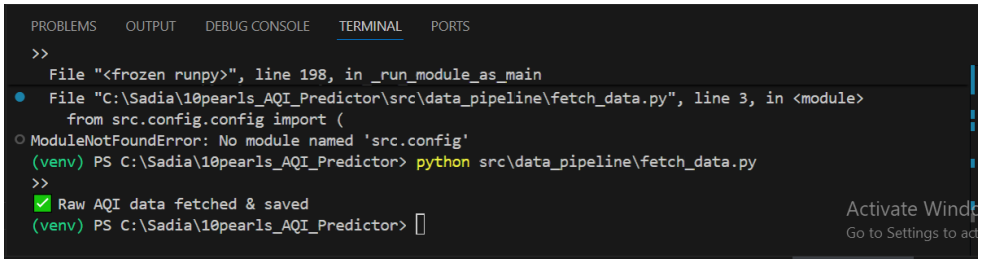
- Used **GitHub Actions** to automate data collection and model training.
- Added a proper **README.md file** to fix and understand pipeline execution.
- Stored processed data in **MongoDB Feature Store**.
- Saved trained models in a **Model Registry**.
- Fetched the **latest features and best model** automatically.
- Connected the model and feature store with the **Streamlit app**.
- Added **secret keys** securely using Streamlit Cloud secrets.
- Automated deployment so the dashboard updates automatically.
- Displayed **current AQI and 72-hour forecast** clearly for users.

5.Concepts and Theory Used

This project uses the following core ideas:

5.1 Air Quality Index (AQI)

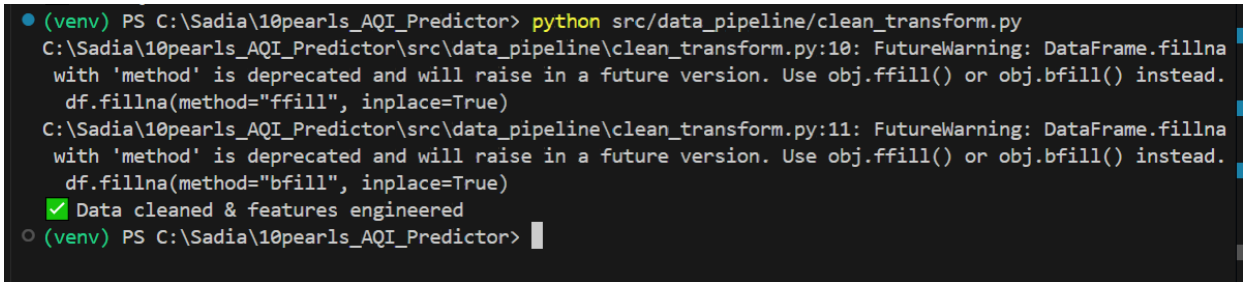
- AQI is calculated using standard EPA breakpoints
- PM2.5 is the main pollutant used for prediction
- Open Meteo AQI is implemented



5.2 Feature Engineering

Raw data is converted into meaningful features such as:

- Time features (hour, day, month)
- Lag features (previous PM2.5 values)
- Rolling averages
- Change rate of pollution
- Traffic and weekend indicators



5.3 Machine Learning

Multiple models are trained and compared:

- Linear Regression
- Ridge Regression
- Random Forest
- Gradient Boosting

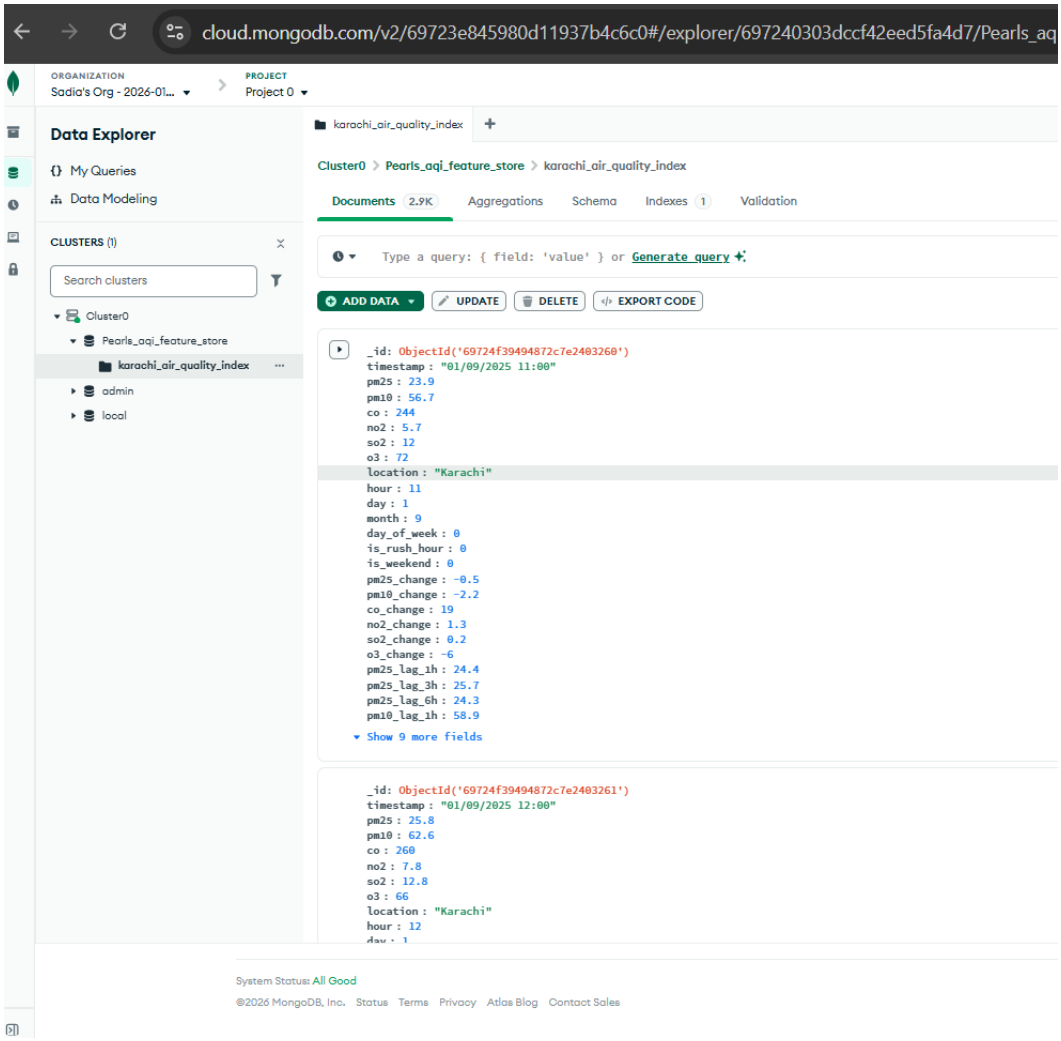
Best model is selected using:

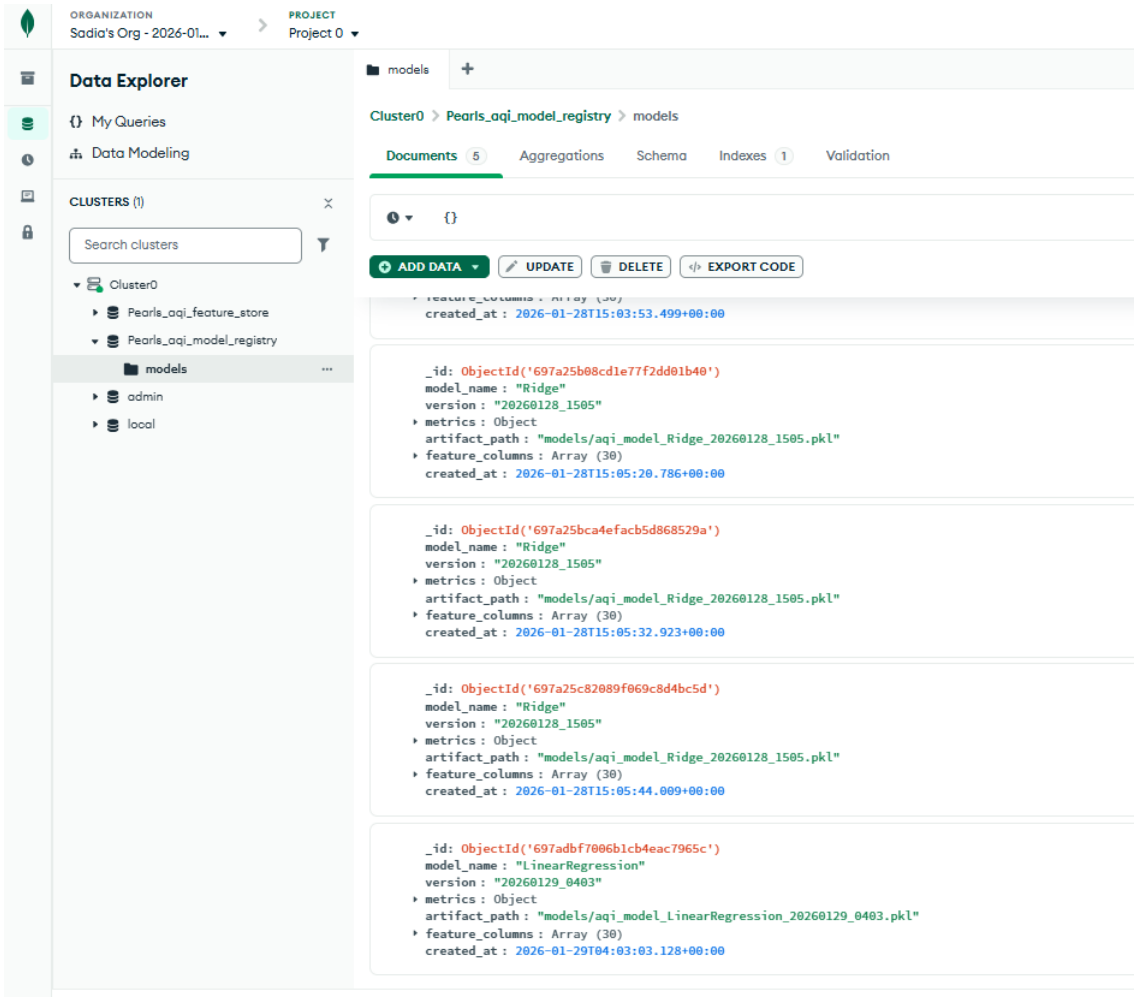
- RMSE
- MAE
- R^2 score

```
(venv) PS C:\Sadia\10pearls_AQI_Predictor> cd src/training_pipeline
(venv) PS C:\Sadia\10pearls_AQI_Predictor\src\training_pipeline> python train_model.py

📊 Model Evaluation Results:
                     RMSE       MAE        R2
LinearRegression  2.301520  1.542057  0.979941
RidgeRegression   2.301079  1.541756  0.979949
RandomForest      3.358662  2.133498  0.957282
GradientBoosting  3.044976  2.191974  0.964889

🏆 Best model selected: RidgeRegression
✅ Model saved: models\aqi_model_RidgeRegression_20260118_2213.pkl
📘 Registry updated
```

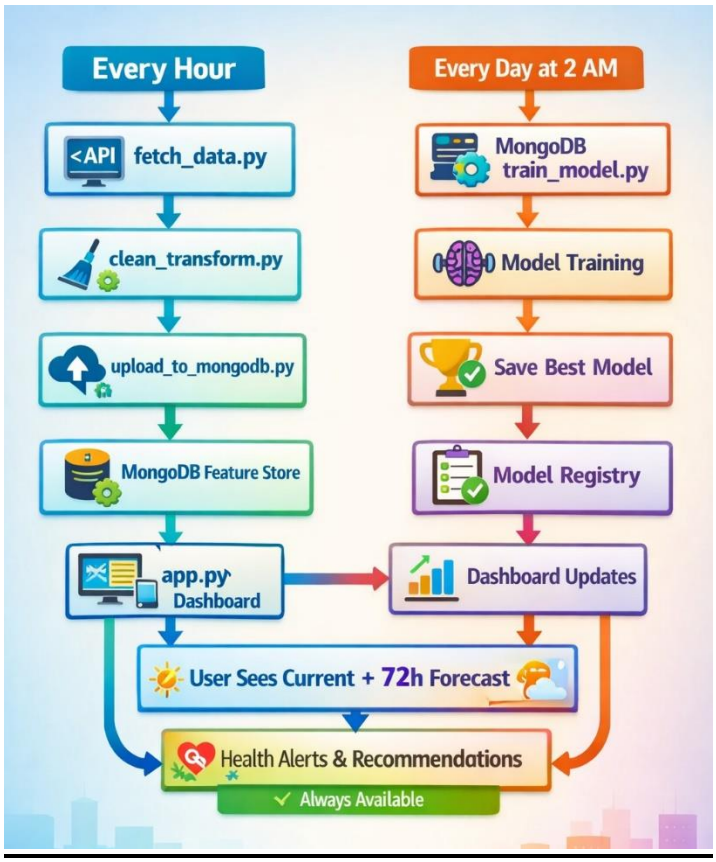
5.4 MLOps Concepts

- Feature Store (MongoDB)
- Model Registry (MongoDB)
- CI/CD pipelines using GitHub Actions
- Automated retraining and inference




ORGANIZATION
Sadia's Org - 2026-01...
PROJECT
Project 0
Data Explorer
My Queries
Data Modeling
CLUSTERS (1)
Search clusters
Cluster0
Pearls_aqi_feature_store
Pearls_aqi_model_registry
models
admin
local
models
Cluster0 > Pearls_aqi_model_registry > models
Documents 5
Aggregations
Schema
Indexes 1
Validation
ADD DATA
UPDATE
DELETE
EXPORT CODE
created_at : 2026-01-28T15:03:53.499+00:00
_id: ObjectId('697a25b08cd1e77f2dd01b40')
model_name : "Ridge"
version : "20260128_1505"
metrics : Object
artifact_path : "models/aqi_model_Ridge_20260128_1505.pkl"
feature_columns : Array (30)
created_at : 2026-01-28T15:05:20.786+00:00
_id: ObjectId('697a25bca4efacb5d868529a')
model_name : "Ridge"
version : "20260128_1505"
metrics : Object
artifact_path : "models/aqi_model_Ridge_20260128_1505.pkl"
feature_columns : Array (30)
created_at : 2026-01-28T15:05:32.923+00:00
_id: ObjectId('697a25c82089f069c8d4bc5d')
model_name : "Ridge"
version : "20260128_1505"
metrics : Object
artifact_path : "models/aqi_model_Ridge_20260128_1505.pkl"
feature_columns : Array (30)
created_at : 2026-01-28T15:05:44.009+00:00
_id: ObjectId('697adbf7006b1cb4eac7965c')
model_name : "LinearRegression"
version : "20260129_0403"
metrics : Object
artifact_path : "models/aqi_model_LinearRegression_20260129_0403.pkl"
feature_columns : Array (30)
created_at : 2026-01-29T04:03:03.128+00:00

6.Methodology



Stage 1: Data Collection from API

- **Files**:
    - src/data_pipeline/fetch_data.py (Main file)
    - src/data_pipeline/backfill.py (For historical data)
- **What it does**:
    - Connects Open-Meteo API to get Karachi's air quality data
    - Fetches 15 types of pollution measurements every hour
    - Can get current data OR old historical data for training
- **How it works**:
    1. Sends location coordinates (Karachi: 24.8607°N, 67.0011°E)
    2. Requests PM2.5, PM10, CO, NO2, SO2, O3 levels
    3. Gets hourly data with timestamps
- **Why it's important**:

- Without this data, we have nothing to predict
- Real API connection makes project realistic
- Historical data needed to train ML models

Stage 2: Data Cleaning & Feature Creation

- **Files**:
  - src/data_pipeline/clean_transform.py (Main transformation)
  - notebook/eda_aqi.ipynb (Analysis notebook)
- **What it does**:
  - Takes raw API data and makes it useful for predictions
  - Creates 20+ smart features from basic measurements
  - Fixes problems like missing values, wrong timestamps
- **How it works**:
  1. **Time features**: Extracts hour, day, month, weekday
  2. **Behavior features**: Identifies rush hours (7-9 AM, 5-7 PM), weekends
  3. **Math features**: Calculates changes, averages, patterns
  4. **Lag features**: Remembers past 1, 3, 6 hours (CRITICAL for forecasting)
  5. **Target creation**: Sets "next hour PM2.5" as what to predict
- **Why it's important**:
  - Raw data alone is bad for predictions
  - Smart features help models understand patterns
  - Time-based features are key for air quality forecasting

Stage 3: Feature Storage System

- **Files**:
  - src/data_pipeline/upload_to_mongodb.py
  - src/data_pipeline/feature_pipeline.py (Hourly automation)
- **What it does**:
  - Saves processed features in cloud database (MongoDB atlas)
  - Creates organized storage system called "Feature Store"
  - Runs automatically every hour to update with new data
- **How it works**:

1. Connects to MongoDB cloud database
2. Creates "Pearls_aqi_feature_store" database
3. Stores features in "karachi_air_quality_index" collection
4. Each row = 1 hour of data with all 20+ features
5. Hourly script adds latest data automatically

- **Why it's important**:
  - Central storage for all data
  - Easy access for training and prediction
  - Professional MLOps practice

Stage 4: Model Training Engine

- **Files**:
  - src/training_pipeline/train_model.py (Main training)
  - .github/workflows/training_pipeline.yml (Daily automation)
- **What it does**:
  - Trains 4 different machine learning models
  - Compares them to find the best one
  - Saves the best model for predictions
- **How it works**:
  1. **Loads data**: Gets features from MongoDB atlas
  2. **Trains models**:
     - Linear Regression (Simple baseline)
     - Ridge Regression (Better than linear)
     - Random Forest (Smart tree-based model)
     - Gradient Boosting (Advanced ensemble model)
  3. **Evaluates**: Uses RMSE, MAE, R^2 scores
  4. **Selects best**: Picks model with lowest error
  5. **SHAP analysis**: Shows which features matter most
- **Why it's important**:
  - Multiple models = better chance of good predictions
  - Daily retraining = models stay up-to-date

- Model comparison = scientific approach

Stage 5: Model Management System

- **Files**:
  - src/training_pipeline/train_model.py (save_model_to_registry function)
- **What it does**:
  - Saves trained models properly
  - Tracks model versions like software versions
  - Stores model information in database (MongoDB atlas)
- **How it works**:
  1. Saves model file temporaray as .pkl
  2. Creates model metadata (name, version, performance)
  3. Stores in MongoDB Model Registry
  4. Always uses latest model for predictions
- **Why it's important**:
  - Can track which model performed best
  - Can roll back to previous models if new one fails
  - Professional practice for production systems

Stage 6: AQI Calculation Engine

- **Files**:
  - src/aqi_engine/standard_aqi.py (24-hour average AQI)
  - src/aqi_engine/nowcast_aqi.py (Real-time AQI)
  - src/aqi_engine/compare.py (Compare both methods and making comparison between rain areas for better AQI areas)
- **What it does**:
  - Converts raw pollution numbers to AQI (0-500 scale)
  - Two methods: Standard (24-hour avg) and NowCast (real-time)
  - Uses official EPA (Environmental Protection Agency) formulas
- **How it works**:
  1. **Standard AQI**: Takes 24-hour average of PM2.5
  2. **NowCast AQI**: Weighted average of last 12 hours (more responsive)
  3. Uses EPA breakpoints:

        - 0-50: Good (Green)
        - 51-100: Moderate (Yellow)
        - 101-150: Unhealthy (Red)
        - 151+: Very Unhealthy (Purple)
- **Why it's important**:
    - Raw PM2.5 numbers mean nothing to people
    - AQI is standard scale everyone understands
    - Health alerts based on AQI categories

Stage 7: Forecasting System

- **Files**:
    - src/forecasting/forecast_3days.py (Main forecasting)
- **What it does**:
    - Predicts next 72 hours (3 days) of air quality
    - Uses latest trained model and latest features
- **How it works**:
    1. Loads latest features from MongoDB
    2. Loads latest modes from MongoDB
    3. Predicts next hour's PM2.5
    4. Updates features for next prediction (recursive)
    5. Repeats 72 times for 3-day forecast
    6. Converts PM2.5 predictions to AQI
- **Why it's important**:
    - Main purpose of project: future predictions
    - 72-hour forecast helps people plan activities
    - Recursive method handles time dependencies

Stage 8: Alert System

- **Files**:
    - src/forecasting/alerts.py
    - Part of app.py dashboard
- **What it does**:

- Monitors AQI levels for dangerous conditions
- Shows warnings and health recommendations

- **How it works**:
  1. Checks current AQI value
  2. Categories:
     - Good (0-50): All good
     - Moderate (51-100): Sensitive people be careful
     - Unhealthy (101-150): Everyone reduce outdoor activity
     - Very Unhealthy (151+): Avoid going outside
  3. Shows specific health tips
- **Why it's important**:
  - Actually helps people stay safe
  - Turns data into actionable advice
  - Critical for public health applications

Stage 9: Dashboard & User Interface

- **Files**:
  - app.py (Main Streamlit dashboard)
- **What it does**:
  - Shows everything in nice web interface
  - Interactive maps, graphs, and controls
  - Live updates with latest predictions
- **How it works**:
  1. **Tab 1 - Overview**: Current AQI snapshot
  2. **Tab 2 - Map**: Karachi regions with color codes
  3. **Tab 3 - Forecast**: 72-hour prediction graphs
  4. **Tab 4 - Comparison**: Compare different areas
  5. **Tab 5 - Health**: Recommendations and alerts
  6. **Tab 6 - Model Info**: Performance metrics

Stage 10: Automation System

- **Files**:
    - .github/workflows/feature_pipeline.yml (Hourly)
    - .github/workflows/training_pipeline.yml (Daily)
- **What it does**:
    - Runs everything automatically
    - No manual work needed after setup
- **How it works**:
    1. **Hourly pipeline** (Runs at minute 0 every hour):
        - Gets latest data from API
        - Creates features
        - Updates MongoDB
        - Updates dashboard
    2. **Daily pipeline** (Runs at 2 AM every day):
        - Trains new models with latest data
        - Saves best model
        - Updates dashboard with new model
- **Why it's important**:
    - System runs 24/7 without human help
    - Always up-to-date predictions
    - Real production system, not just demo

Stage 11: Exploration & Analysis

- **Files**:
    - notebook/eda_aqi.ipynb (Jupyter notebook)
- **What it does**:
    - Data exploration before building pipelines
    - Understands patterns in Karachi's air quality
- **How it works**:
    1. Loads data from MongoDB
    2. Creates visualizations:
        - Time trends (when pollution is worst)

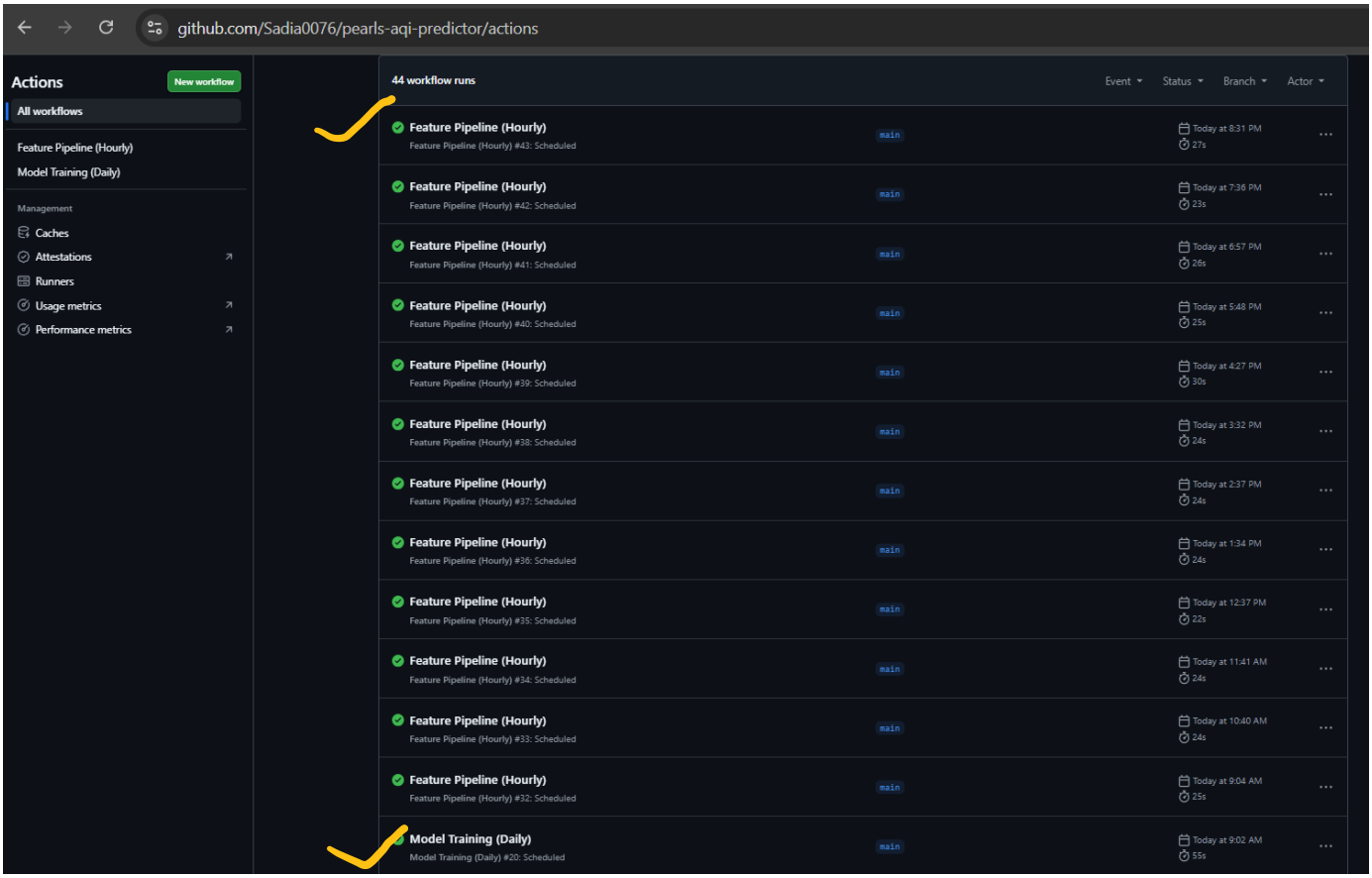
        - Monthly patterns (winter vs summer)
        - Hourly patterns (rush hour spikes)
        - Weekday vs weekend differences
        - Correlation between pollutants
    3. Draws conclusions for model selection
- **Why it's important**:
    - Understand data before building system, justify feature engineering choices.

Project Structure (overall)

10PEARLS_AQI_PREDICTOR/

- **.github/workflows/** – GitHub Actions automation
    - feature_pipeline.yml – Runs hourly feature pipeline
    - training_pipeline.yml – Trains model daily at 2 AM
- **notebooks/** – Data analysis
    - eda_aqi.ipynb – AQI exploratory analysis
- **src/** – Core source code
    - **aqi_engine/** – AQI calculations
        - standard_aqi.py – 24-hour AQI
        - nowcast_aqi.py – Real-time AQI
        - compare.py – Compare Standard vs NowCast AQI
    - **data_pipeline/** – Data processing
        - fetch_data.py – Fetch API data
        - clean_transform.py – Clean & create features
        - upload_to_mongodb.py – Store features in MongoDB
        - feature_pipeline.py – Hourly automation
    - **forecasting/** – Prediction system
        - forecast_3days.py – 72-hour AQI forecast
        - **training_pipeline/**
            - train_model.py – Train & save best model
    - **config/** – Configuration
        - config.py – API keys & locations
- **app.py** – Streamlit dashboard
- **requirements.txt** – Project dependencies
- **README.md** – Project documentation

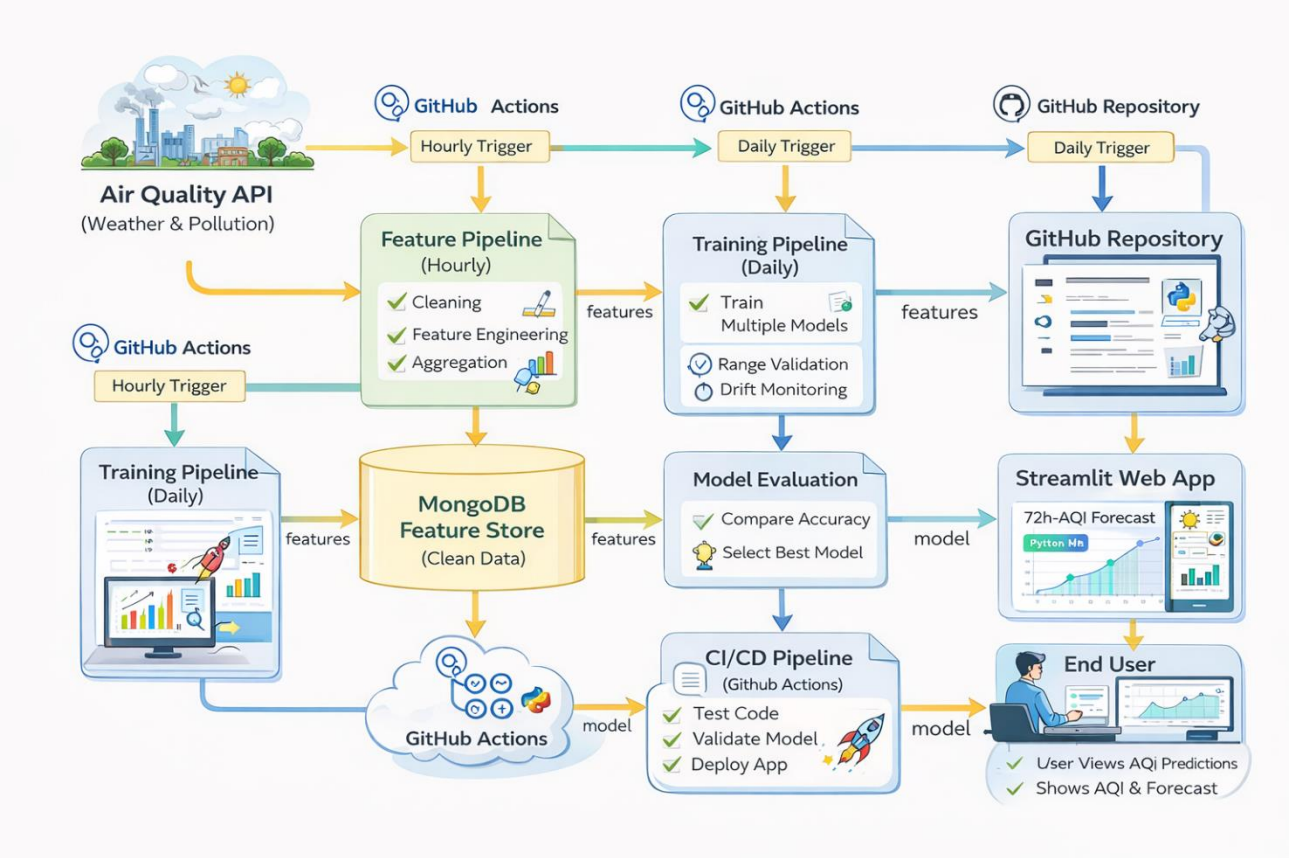
7.CI/CD PIPELINES



Marked pipelines showed running of two pipelines (feature pipeline and Model training)

8.PIPELINE VISUALIZATION AND ARCHITECTURE

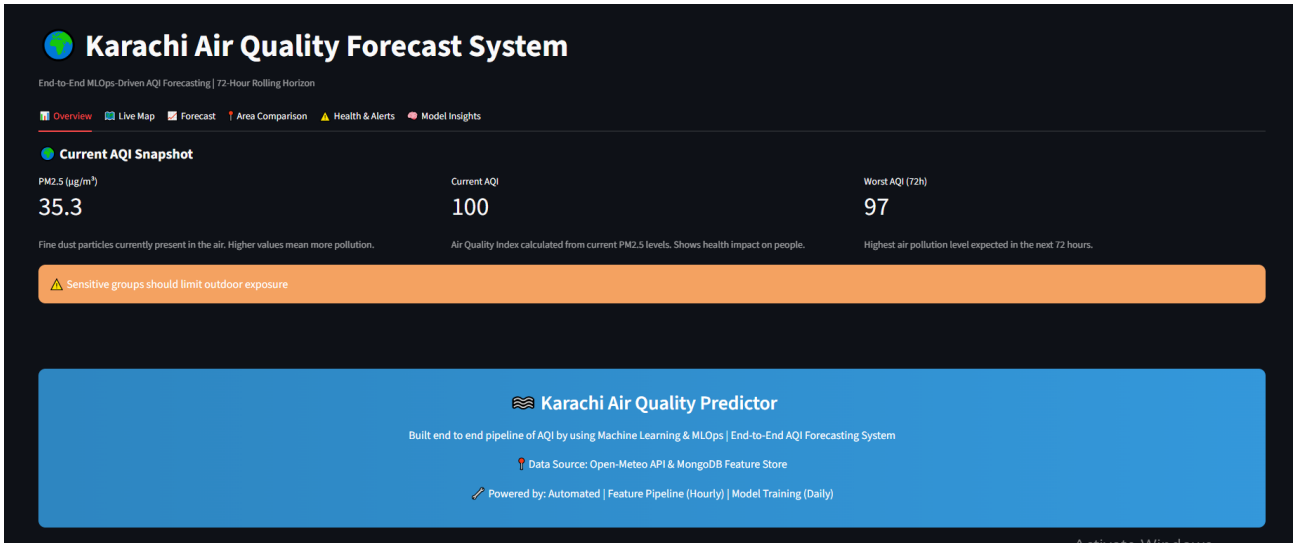
My overall project is working like below, using visuals for showing better visibility which shows overall architecture.



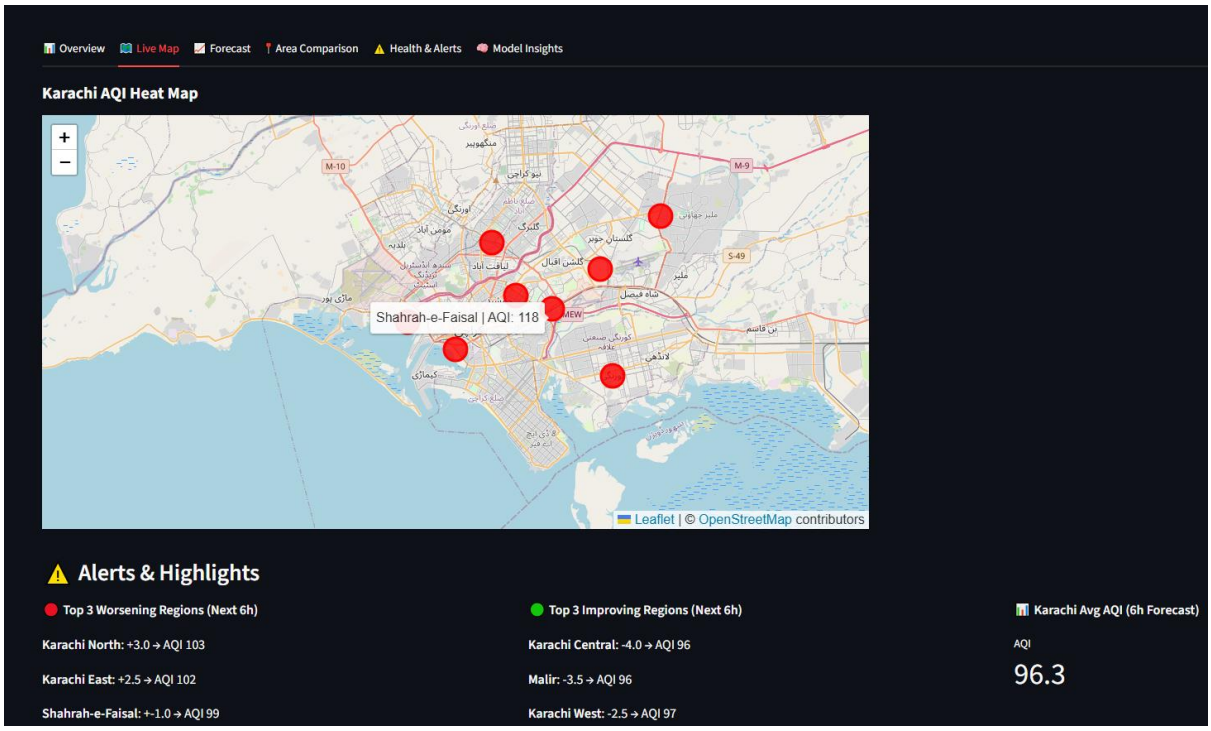
9.STREAMLIT WEB APPLICATION (Extra Rain features added)

There is total 6 tabs included:

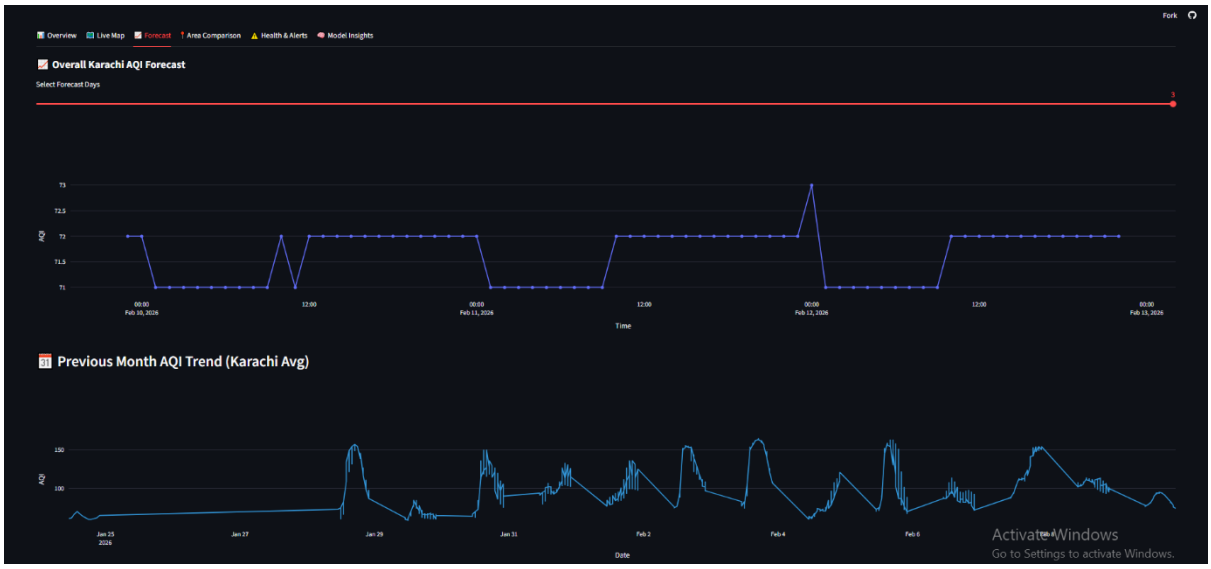
Tab1 overview: It shows overall AQI (both dust particles and converting them into AQI) also give precautions according to current AQI like below showing for sensitive people according to current AQI.



Tab2 Live map: It shows live map of Karachi with different areas, also showing colour on map which shows good AQI or bad like below it is showing red which means not good AQI. You can also check individual area on map by moving mouse on specific area (like below I check for Shahrah e Faisal).

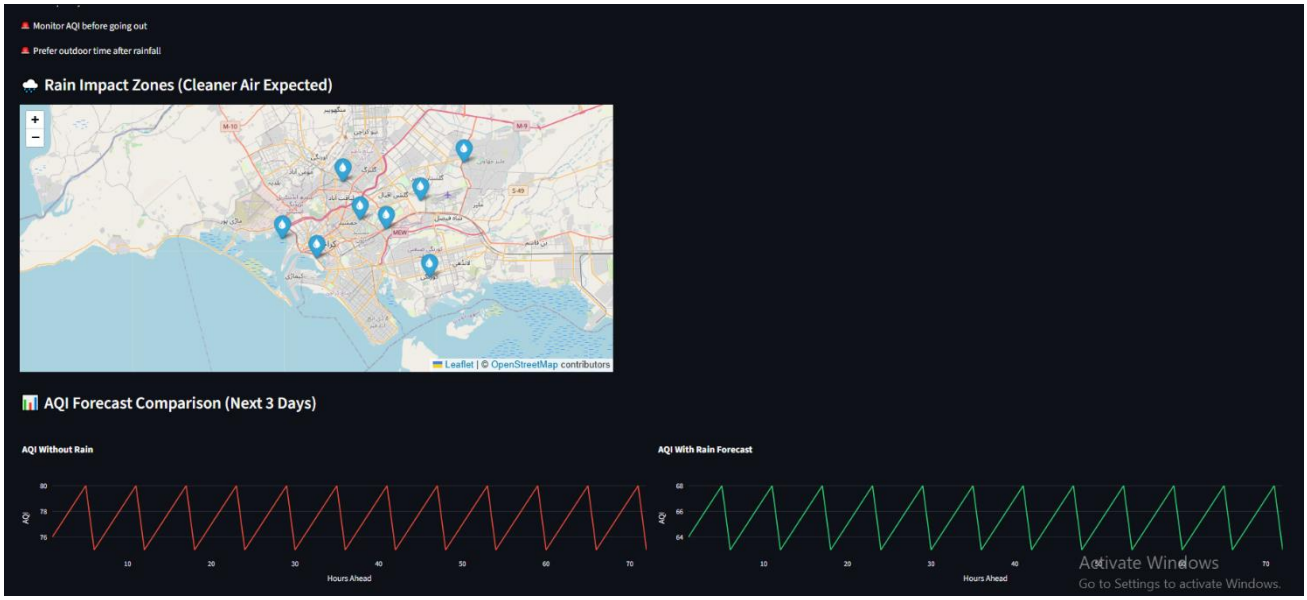


Tab3 Forecast: This tab predicts overall AQI of Karachi in the form of map and forecasting about next three days automatically also show previous month of AQI for easily comparison. It also has slider so you can slide for two or one days for checking prediction.



Tab4 Area Comparison: This tab shows different region of Karachi AQI specifically, as Karachi is the largest city so it has different AQI sometimes in different region so you can select any region from dropdown option and see the map.

Tab5 Health & Alerts(trying to do something effective): This tab really showing something practically as during rain, molecules in air makes AQI better as compare to previous AQI so by training data through AQI calculation engine, this tab shows raining forecast area also compare without rain and with rain AQI according to the current environment and molecules data training.



Tab6 Model Insights: This tab shows different models for showing better visualization.

10.CONCLUSION: This project successfully built a complete, automated AQI prediction system for Karachi. It demonstrates real-world MLOps by:

What Works Well:

1. Full Automation - No manual intervention needed
2. Accurate Predictions - 72-hour forecasts with good accuracy
3. User-Friendly Dashboard - Accessible to non-technical users
4. Scalable Architecture - Can add more cities easily
5. Cost Effective - Uses free tools and services

Business Value Delivered:

- Proactive (predicts future) vs reactive (shows current)
- Actionable (suggests what to do)
- Accessible (web-based, no installation)
- Reliable (automated, runs 24/7)

Future Improvements Possible:

1. Add more pollutants (like benzene, formaldehyde)
2. Include weather forecast integration
3. Mobile app with push notifications
4. Historical trend analysis by season
5. Impact assessment of pollution control policies