# Detailed Report on Air Quality Index (AQI) Prediction System

## Overview of the Project

This project focuses on building an end-to-end Air Quality Index (AQI) prediction system for the next three days. The system integrates various components, including data ingestion, processing, machine learning model training, and a web application for visualization and prediction. It leverages modern tools and frameworks, such as Hopsworks for feature storage, FastAPI for backend development, and Streamlit for the frontend dashboard. The goal was to create a scalable, automated pipeline for AQI forecasting while achieving high prediction accuracy.

## APIs Used

Two external APIs were utilized in this project:  
  
1. Open-Meteo API: To fetch weather data, including temperature, precipitation, and wind speed.  
2. OpenWeather API: To retrieve air pollution data, such as AQI and various pollutant concentrations (e.g., CO, NO2, PM2.5).  
  
These APIs provided the foundational raw data for feature extraction and prediction.

## Key Achievements

### 1. Feature Engineering and Data Storage

- Feature Pipeline Script:  
 - Created a Python script to fetch raw weather and pollutant data from the APIs.  
 - Computed features such as time-based variables (hour, day, month), AQI change rate, and pollutant concentrations.  
 - Stored the processed features in the Hopsworks Feature Store.  
  
- Backfilling Historical Data:  
 - Developed another script to backfill 400 days of historical data into the feature store.  
 - This ensured a robust dataset for model training and evaluation.

### 2. Model Training

- Loaded the historical data from Hopsworks and performed Exploratory Data Analysis (EDA) to identify trends and correlations in the data.  
- Experimented with multiple machine learning models:  
 - Linear Regression  
 - Lasso Regression  
 - Ridge Regression  
 - Random Forest Regressor  
  
- Results of Model Evaluation:  
 - Ridge Regression outperformed Linear and Lasso Regression.  
 - Random Forest delivered the best performance with an accuracy of 97%.  
  
- Final Model: Trained and saved the Random Forest model in the Hopsworks Model Registry for deployment.

### 3. Automation with CI/CD

- Designed a CI/CD pipeline using GitHub Actions to:  
 - Run the feature script hourly to update data.  
 - Execute the model training script daily to keep the model updated with new data.

### 4. Data Forecasting

- Integrated the forecast APIs to fetch weather and air pollution data for the next three days.  
- Processed the forecasted data and saved it as a CSV file for future use and analysis.

### 5. Web Application Development

- Backend:  
 - Developed the backend using FastAPI to serve the prediction results.  
 - Implemented endpoints for fetching forecast data and AQI predictions.  
  
- Frontend:  
 - Built a simple and interactive dashboard using Streamlit.  
 - The dashboard allows users to view the predicted AQI for the next three days in a user-friendly format.

## Project Workflow

1. Data Ingestion:  
 - Fetch raw data from Open-Meteo and OpenWeather APIs.  
 - Process and store the data in Hopsworks Feature Store.  
  
2. Model Training:  
 - Load historical data from the feature store.  
 - Train and evaluate multiple machine learning models.  
 - Deploy the best-performing model (Random Forest) to the Hopsworks Model Registry.  
  
3. Automation:  
 - Implement a CI/CD pipeline for automating data updates and model retraining.  
  
4. Forecasting and Visualization:  
 - Fetch forecast data for the next three days using the APIs.  
 - Predict AQI and display the results on the Streamlit dashboard.

## Tools and Technologies Used

- APIs: Open-Meteo, OpenWeather  
- Data Storage: Hopsworks Feature Store  
- Machine Learning: Scikit-learn (Random Forest, Ridge Regression, etc.)  
- Backend: FastAPI  
- Frontend: Streamlit  
- Automation: GitHub Actions

## Key Results

- Achieved 97% accuracy using the Random Forest model for AQI prediction.  
- Developed a fully functional web application to visualize AQI predictions for the next three days.  
- Automated the data pipeline for continuous updates and model retraining.

## Future Enhancements

- Scalability: Containerize the application using Docker for easier deployment.  
- Explainability: Use SHAP or LIME to provide insights into feature importance.  
- Alerts: Implement notifications for hazardous AQI levels.  
- Advanced Models: Experiment with deep learning models for improved forecasting accuracy.

## Conclusion

This project successfully delivers an end-to-end AQI prediction system with a high degree of accuracy and automation. By integrating feature engineering, machine learning, and web application development, it provides a comprehensive solution for real-time AQI forecasting and visualization. The modular design ensures scalability and maintainability, making it a valuable tool for monitoring air quality in any city.

## Dashboard



