**AQI Prediction Project**

Data Science AI/ML

Final Report Submission by

**Mehnoor Fayyaz**

**Software Engineering Student**

Data Science 10P Shine Intern

[fayyazmehnoor@gmail.com](mailto:fayyazmehnoor@gmail.com)

Submission Date

02/08/2025

Internship Duration

8 weeks

4.6

***Model Registry***

***Model Training and Evaluation***

4.5

4.2

4.3

4.4

***Tools and Technologies***

***Project Overview***

4

2

4.1

***Visualization with Gradio***

1

3

***Project Description***

***Objectives***

***Data Collection***

***Automation with GitHub Actions***

***Feature Store Integration***

***Table of Contents***

Final Project Report

# Project Overview

The project overview was to predict Air Quality Index of Lahore for the next 3 days and display the final predictions in a descriptive dashboard.

1. Objectives

Key objectives that I followed for this project were:

* To collect real-time AQI and weather data using external APIs.
* To automate data collection and model training using a CI/CD pipeline.
* To implement and evaluate multiple regression models for AQI prediction.
* To store and manage features and models using a feature store.
* To visualize predictions through an interactive web application.

1. Tools & Technologies

Tools and technologies that I used and learned during this internship:

* Python
* Pandas, Matplotlib, Seaborn and Scikit-Learn.
* GitHub Actions
* Feature Store (Hopsworks)
* Gradio
* Hugging Face

1. Project Description

The primary objective of this project was to build a machine learning-based solution for predicting the **Air Quality Index (AQI)** of Lahore using real-time environmental and weather data. The project involved multiple stages, including data collection, preprocessing, model training, automation, and visualization. Key steps followed during the development:

## **Data Collection:**

Raw data related to AQI and pollutants such as PM2.5, PM10, NO₂, NO, CO, O3 and SO₂ was collected from the **Open Weather API**, while relevant weather data like temperature, wind speed, humidity and AQI value was fetched from the **IQAir API**. This data is being collected since July at an **hourly interval** to ensure up-to-date model inputs.

## **Automation with GitHub Actions:**

A CI/CD pipeline was implemented using **GitHub Actions** to automate the entire process. The pipeline collects new data every hour, appends it to a **CSV file**, and runs the model training script **daily** without manual intervention.

## **Feature Store Integration:**

The **Hopsworks Feature Store** was explored to some extent and utilized to manage the daily CSV files containing the extracted features.

## **Model Training and Evaluation:**

1. First Approach

In my initial approach, I began by collecting data on pollutants such as SO₂, NO₂, NO, O₃, PM₂.₅, and PM₁₀, along with the categorical AQI values (1, 2, 3, 4, 5) from the **OpenWeather** API. After a week of data collection, I realized that the project required realistic, continuous AQI values rather than categorical ones. This led me to restart the process and collect data from scratch using the **IQAir** API.

From IQAir, I gathered weather parameters such as wind speed, temperature, and humidity, along with accurate AQI values that aligned with the project requirements. After several weeks of continuous data collection, I moved on to the data cleaning and analysis phase using Pandas, where I learned how to handle **missing values** and ensure the dataset was properly prepared before training the model.

To automate the data collection process, I implemented **a CI/CD pipeline** using GitHub Actions, allowing the data to be recorded hourly into a CSV file. I then began exploring machine learning regression models, starting with Random Forest Regression, to predict AQI directly from the pollutant and weather data. However, I soon realized that attempting to build the model without first performing thorough Exploratory Data Analysis (EDA) was a significant oversight in my approach.

1. Second Approach

In my second approach, I understood the importance of **performing Exploratory Data Analysis (EDA)** before adding or selecting features. At this stage, I began learning how to visualize pollutant and weather data using Matplotlib and Seaborn.

* I created **regression plots** (regplots) for each pollutant and weather parameter against the AQI values, which displayed a regression line passing through the scatter points.
* I also used **histograms** to examine the distribution of AQI values.
* I applied a **heatmap** to analyse the correlation between AQI and all collected pollutants and weather parameters. Through this process, I learned that correlation values range from -1 to +1, where values close to zero indicate weak or no relationship.

Using the correlation insights, I identified the variables with strong positive or negative correlations to AQI. Based on this analysis, I selected NO₂, O₂, CO, NO, temperature, and humidity as the most relevant features for predicting AQI.

I then trained a Random Forest Regressor model using these features. However, the performance metrics of this model were not satisfactory, with an **R² score** of 0.67, and **RMSE** of 25.56.

1. Third Approach

In my third approach, I explored the Ridge Regression model, which is particularly useful when independent variables are highly correlated, as it helps prevent unstable and unreliable regression coefficients.

I applied **Ridge Regression** and compared its performance to the Random Forest Regressor. The R² score for Ridge Regression was significantly higher, reaching approximately 0.95, outperforming the Random Forest model.

At this stage, I shifted my focus from predicting AQI values directly to predicting PM₂.₅ concentrations, which could then be used to calculate AQI if needed.

During this phase, I also explored the concept of **time series feature engineering.**

* **Lag features:** which represent past values of a time series and can be included as input variables in forecasting models
* **Rolling windows**: a method in which a fixed-size window moves across the dataset to compute statistics such as the mean, minimum, and maximum.
* **Expanding windows**: where the training dataset grows over time as new data becomes available.

I began comparing the R² scores for training and testing datasets in Ridge Regression. The model achieved an R² score of 0.97 on the training data and 0.95 on the testing data, suggesting strong performance.

During this process, I learned about two important concepts:

* **Overfitting**: where a model performs well on training data but poorly on testing data.
* **Underfitting**: where the model performs poorly on both.

However, when I compared the predicted PM₂.₅ values to the actual test values, they appeared almost identical, raising suspicion about the results. This prompted me to research the concept of **data leakage**, which occurs when information from the test set influences the training process, leading to unrealistically high-performance metrics.

1. Fourth Approach

In my fourth approach, I addressed the issue of **data leakage**, which occurs when test data is inadvertently included in the training process and influences model performance. To eliminate this problem, I separated the training and testing datasets properly.

Specifically, I reserved the last 72 rows of my dataset as the testing **set (test\_df)** and used all remaining rows as the training set **(train\_df).** This ensured that the model was evaluated only on unseen data. However, despite this adjustment, the performance of the Ridge Regression model remained unchanged, and the predictions were still very close to the actual values.

I again used Random Forest Regressor this time to predict PM₂.₅ but the result of R2 score were still near to 0.76.

With these concepts in mind, I began exploring the XGB Regressor as a potential next step in improving model performance.

1. Final Approach

In my final approach, I implemented the **XGBoost Regressor**. XGBoost regression is a machine learning algorithm used for predicting a continuous target variable.

It incorporates regularization terms to control the complexity of the individual trees and the overall model, which helps in preventing overfitting and improving generalization.

I applied it to the prepared dataset, incorporating the relevant pollutants and weather parameters, along with the engineered time series features.

## **Exploratory Data Analysis:**

Exploratory Data Analysis was performed to identify features with a strong correlation to the Air Quality Index (AQI). Visual techniques such as **scatter plots**, and **correlation heat maps** were used to analyze the relationships between pollutants and weather parameters, with pm2.5.

Based on this analysis, the following features were selected for further processing:

* ***NO₂****,* ***O₃****,* ***CO****,* ***NO*** (pollutants)
* **Humidity and Temperature** (weather parameters)

1. **Feature Engineering**

Additional features were engineered to capture **trends** in pm2.5 values. These features were designed to incorporate recent historical pm2.5 patterns into the model. Their inclusion significantly improved model performance and raised the R2 score and adjusted R2 score as well. The following lag-based features and rolling windows were created:

* **pm2.5\_lag1**: pm2.5 value from the previous hour.
* **pm2.5 \_lag2**: pm2.5 value from two hours prior.
* **pm2.5 \_avg3**: Average of the last three hours.
* **pm2.5 \_avg7: Average of last 7 hours.**
* **pm2.5 \_std7: Standard Deviation of last 7 hours.**

Some important steps taken:

* After fetching data from Hopsworks, the whole data was shuffled unexpectedly and time series data should not be shuffled so I managed to sort the data before applying ML model on it.
* I dropped all the Nan values after feature engineering of time series data.
* I separated the training data and testing data to avoid data leakage; so that no information of testing dataset reaches in training dataset.
* I used last 72 rows of testing data to predict the next 72 hours pm2.5.
* I learnt about **log transform** to remove outliers from the parameters for normalization but I generally did not use it.
* I also used **adjusted R2** score to remove irrelevant features and I removed all the irrelevant features like , pm2.5\_avg14 and pm2.5\_max.

**Adjusted R2 score** = 1-[]

1. **Regression Metrices**

|  |  |
| --- | --- |
| REGRESSION METRICES | |
| MSE: 19.097 | **Training R2 score**:0.98 |
| MAE: 3.369 | **Testing R2 score**: 0.901 |
| RMSE: 4.37 | **Adjusted R2 Score:** 0.86 |

**(RMSE/mean of actual values) \*100**

* rmse\_train: 4.564863373023357
* rmse\_test: 6.2545397907103695
* accuracy train: 6.200362567256713
* accuracy test: 8.502187534100113

1. **EPA-US AQI Calculation Formula:**

After predicting pm2.5, I used the predicted concentrations of pm2.5 and calculated AQI using a formula given by Environmental Protection Agency of United States of America.

**AQI**=((AQI\_high-AQI\_low)\*(i-BP\_low))/(BP\_high-BP\_low)+AQI\_low

* **BP\_High** = Highest breakpoint concentration of pm2.5
* **BP\_Low** = Lowest breakpoint concentration of pm2.5
* **AQI\_High** = Highest AQI value
* **AQI\_Low** = Lowest AQI value
* **i** = predicted concentration of pm2.5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Breakpoint Low** | **Breakpoint High** | **AQI Low** | **AQI High** | **AQI Category** |
| 0.0 | 12.0 | 0 | 50 | Good |
| 12.1 | 35.4 | 51 | 100 | Moderate |
| 35.5 | 55.4 | 101 | 150 | Unhealthy |
| 55.5 | 150.4 | 151 | 200 | Unhealthy |
| 150.5 | 250.4 | 201 | 300 | Unhealthy |
| 250.5 | 350.4 | 301 | 400 | Hazardous |
| 350.5 | 500.4 | 401 | 500 | Hazardous |

For example; if the predicted concentration of pm2.5 was **97.3** then its BP\_low would be 55.5, BP\_high would be 150.4, AQI\_high would be 151 and AQI\_low would be 200.

## **Model Registry:**

Once the model was trained, the model metrics, pm2.5 predictions and calculated AQI were stored in the **model registry** of the feature store for future use and deployment.

## **Visualization with Gradio:**

The final predictions were visualized using **Gradio**. The Gradio app was deployed on **Hugging Face Spaces**, allowing real-time visualization of both the model predictions and the historical data trends of AQI.

Dashboard includes:

* Next 3 days **maximum** pm2.5 prediction in a data frame.
* Data frame for alerts like Good, Moderate, Unhealthy and Hazardous AQI.
* **Hourly pm2.5** prediction of next 72 hours in a line chart.
* **Hourly AQI** calculated of next 72 hours in a line chart.
* Data frame for hourly pm2.5 and calculated AQI prediction for readability.

Kindly examine the dashboard available on this link:

<https://huggingface.co/spaces/Mehnoor00/Lahore-AQI-Predictions>