**AQI PREDICTION SYSTEM REPORT**

**1. Introduction**

Air Quality Index (AQI) is a crucial metric for assessing the quality of air in a given area. This report outlines the development and implementation of an AQI prediction system that utilizes various data sources, machine learning techniques, and a feature store for efficient data management.

**2. Objectives**

The primary objectives of the AQI prediction system are:

* To predict future AQI values based on historical air quality and weather data.
* To provide insights into the factors affecting air quality.
* To facilitate timely alerts for poor air quality conditions.

**3. Data Sources**

The system integrates data from multiple sources:

* **Air Pollution Data**: Historical AQI data was fetched using the OpenWeatherMap API.
* **Weather Data**: Weather data was obtained using the Open-Meteo API, which provides hourly weather metrics such as temperature, humidity, and precipitation.

**4. Data Collection**

**4.1 Air Pollution Data**

The air pollution data was collected using the following function:

1def fetch\_air\_pollution\_data(lat, lon, start, end, api\_key):

2 # API call to fetch air pollution data

**4.2 Weather Data**

Weather data was collected using the Open-Meteo API:

1def fetch\_weather\_data(lat, lon, start\_date, end\_date):

2 # API call to fetch weather data

**5. Data Preprocessing**

The collected data underwent several preprocessing steps:

* **Datetime Conversion**: Timestamps were converted to datetime objects for easier manipulation.
* **Feature Engineering**: New features such as day, month, and year were extracted from the datetime column.
* **Aggregation**: Daily averages of AQI and weather metrics were calculated to create a consolidated dataset.

**6. Feature Engineering**

The following features were included in the model:

* Air quality metrics (e.g., AQI, PM2.5, PM10, CO, NO2, etc.)
* Weather metrics (e.g., temperature, humidity, precipitation)
* Temporal features (e.g., day, month, year)

**7. Model Development**

**7.1 Model Selection**

A Gradient Boosting Regressor was chosen for the prediction task due to its effectiveness in handling regression problems with complex relationships.

**7.2 Model Training**

The model was trained using the following code:

1model = GradientBoostingRegressor(n\_estimators=100, learning\_rate=0.1, max\_depth=6, random\_state=42)

2model.fit(X\_train, y\_train)

**7.3 Model Evaluation**

The model's performance was evaluated using Mean Squared Error (MSE) and R² Score:

1y\_pred = model.predict(X\_test)

2mse = mean\_squared\_error(y\_test, y\_pred)

3r2 = r2\_score(y\_test, y\_pred)

**8. Predictions**

The model was used to predict AQI values for the next three days. The predictions were displayed as follows:

1for i, date in enumerate(future\_dates):

2 print(f"Predicted AQI for {date.date()}: {future\_predictions[i]}")

**9. Feature Store Integration**

The features were stored in a Hopsworks feature store for efficient management and versioning. The following code was used to insert the features into the feature group:

1fg.insert(merged\_dataset)

**10. Conclusion**

The AQI prediction system successfully integrates air quality and weather data to predict future AQI values. The use of machine learning techniques and a feature store enhances the system's efficiency and scalability. Future work may include:

* Incorporating additional data sources (e.g., traffic data, industrial emissions).
* Implementing real-time prediction capabilities.
* Developing a user interface for better accessibility of predictions and insights.

**11. References**

* OpenWeatherMap API Documentation
* Open-Meteo API Documentation
* Hopsworks Feature Store Documentation
* Scikit-learn Documentation