Apache Kafka for Real-time Air Quality Monitoring and Prediction

Introduction

Air quality monitoring is vital in environmental conservation and management of public health. Traditional monitoring systems suffer from latency in data processing, rendering real-time analysis a challenge. In this project, Apache Kafka, an event streaming platform designed to run in distributed systems, is utilized to handle air quality data in real-time and develop predictive models.

UCI Air Quality data contains hourly samples of various pollutants such as CO, NOx, NO2, and Benzene for an Italian city from the years 2004-2005. Through the use of Kafka's feature to stream data, we can simulate real-time flow and build models which predict concentrations of pollutants based on past history and weather conditions.

The objective is twofold: one is to build a robust data pipeline using Kafka for the processing of streaming sensor data; the other is to develop and test forecasting models that can forecast pollutant concentrations. This solution demonstrates how streaming technologies can enhance environmental monitoring systems by enabling real-time data processing and predictive analysis.

Kafka Setup Description

Installation and Configuration

Apache Kafka installation took some steps, beginning with the installation of Java Development Kit 17 (JDK17) using Homebrew. After verification of Java installation using `java -version`, I downloaded the latest Apache Kafka (version 3.6.1) binary package and extracted it. The intention was to utilize KRaft mode (the newconsensus protocol that has replaced ZooKeeper), but this required the generation of a cluster ID. This generated a UUID that needed to be added to the server configuration. I did experience aproblem with the `meta.properties` file, which caused cluster ID inconsistency.

Resolving Configuration Issues

To fix this issue, I had to:

1. Locate the `meta.properties` file in the Kafka logs directory (specified by `log.dirs` in `server.properties`)
2. Remove the existing `meta.properties` file.
3. Initialize the storage using the newly generated cluster ID.

By following these steps, I was able to run the Kafka server by bin/kafka-server-start.sh config/kraft/server.properties

With Kafka up and running, I created a topic exclusively for air quality data named as air-quality and verified its presence.

Producer and Consumer Implementation

For Python-based interaction with Kafka, I installed the necessary package of kafka-python

The producer implementation (`producer.py`) was designed to:

1. Load the UCI Air Quality dataset
2. Process and clean the data (handling -200 values as missing data)
3. Parse date/time fields into proper datetime objects
4. Send records to the Kafka topic with a time delay that simulates real-time generation
5. Include comprehensive error handling and logging

The consumer implementation (`consumer.py`) was designed to:

1. Connect to the Kafka topic
2. Process incoming messages
3. Store data in an SQLite database for subsequent analysis
4. Include robust error handling and logging

Data Exploration Findings

Dataset Overview

The UCI Air Quality data contains hourly samples of a variety of pollutants in an Italian city station. Preliminary investigation underscored certain critical features:

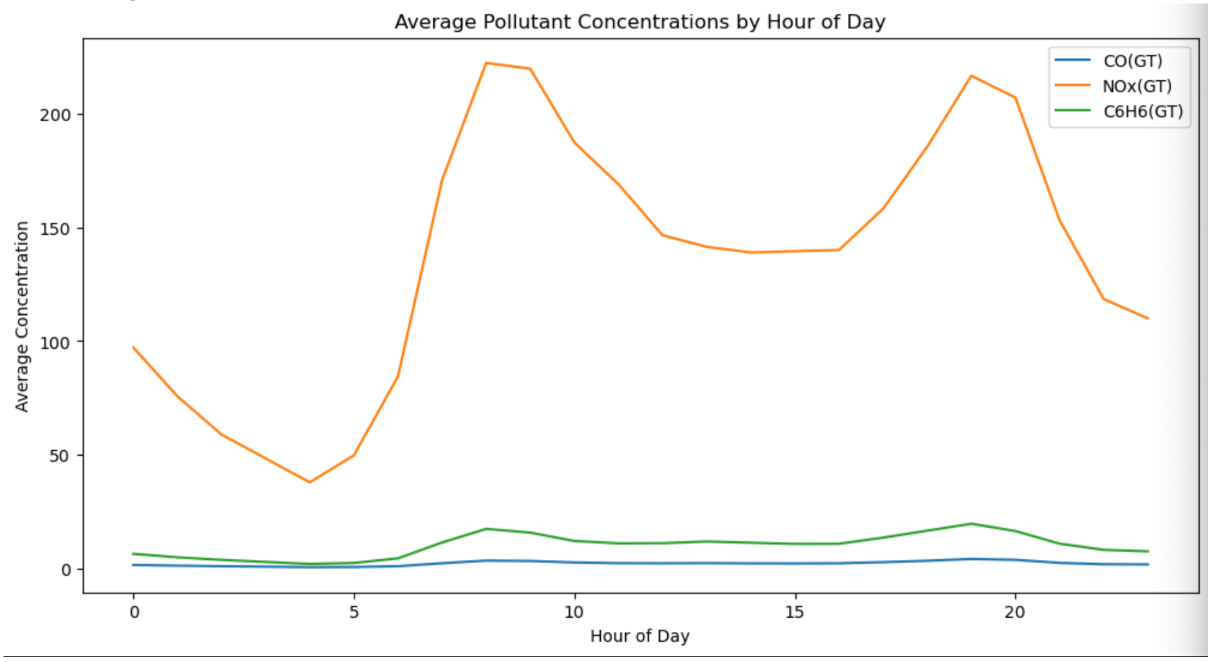
1. Time Series Character: March-April 2004, hour-wise samples
2. Missing Values: Around 15% of values with -200listed as missing
3. Important Variables: CO, NOx, NO2, Benzene, Temperature, Relative Humidity

Temporal Patterns

Time-series analysis revealed unambiguous patterns in the levels of the pollutants:

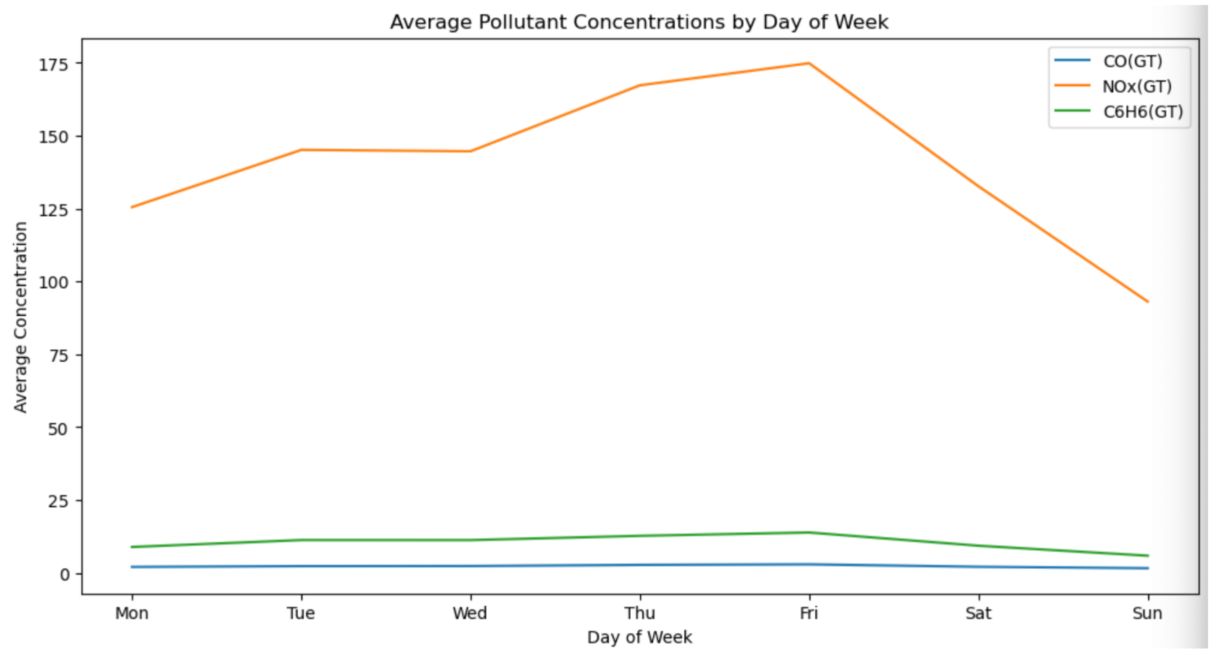
Daily Cycles

- CO and NOx had clear diurnal patterns with maxima around morning (7-9 AM) and evening (6-8 PM) rush hours
- The pattern corresponds to traffic flows, and NOx concentrations 2-3× higher at rush hours



Weekly Patterns:

- Weekday levels were consistently higher than weekend levels
- Friday had the highest levels of pollution for all contaminants reported
- Sunday had the lowest levels (20-30% below weekday means)

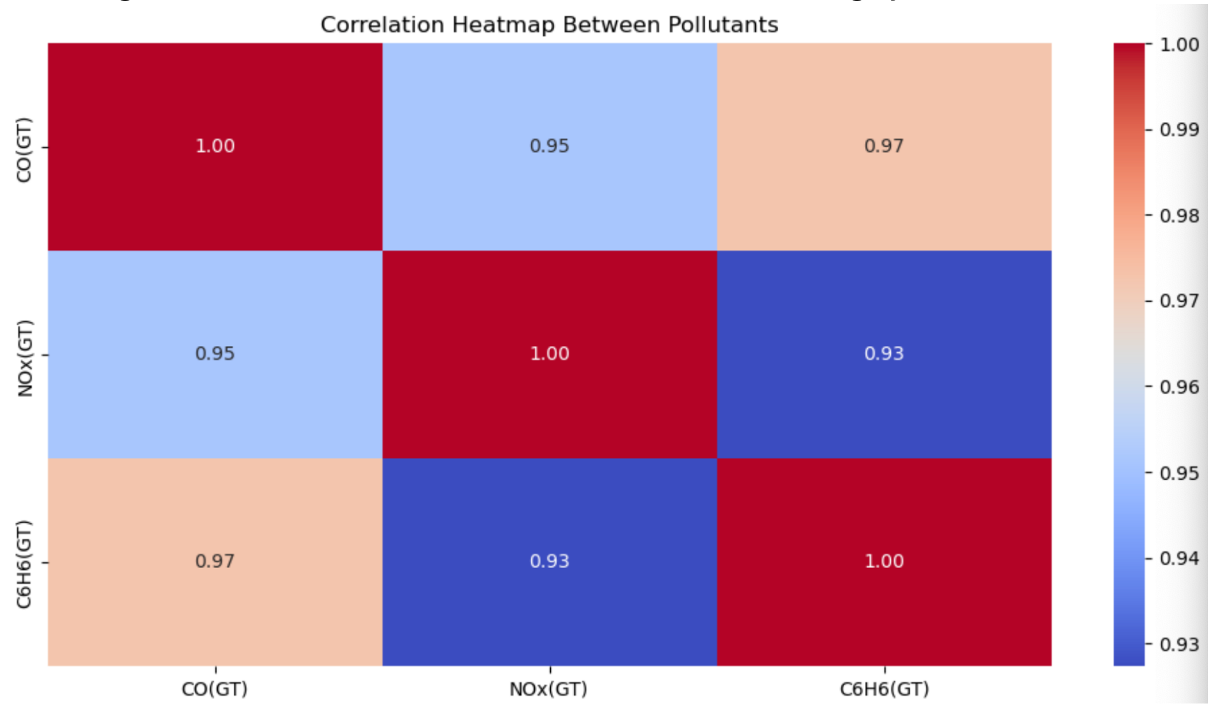


<u>Correlation Analysis</u>

The heatmap graph revealed very high correlations between pollutants:

- CO and NOx: 0.95 correlation
- CO and Benzene: 0.97 correlation
- NOx and Benzene: 0.93 correlation

These high correlations reflect common emission sources, largely vehicular traffic

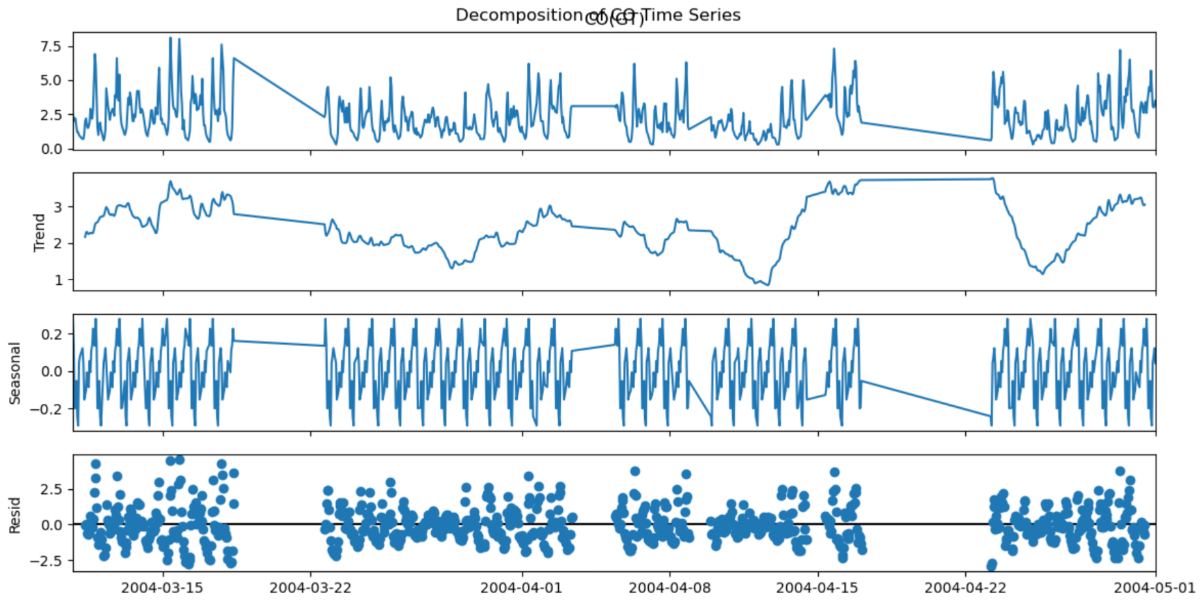


<u>Seasonal Decomposition</u>

Decomposition of CO time series revealed:

- Trend Component: Stronger longer-term fluctuations with high values in mid-March and mid-April

- Seasonal Component: Clear daily cyclical patterns with highly regular oscillations
- Residual Component: Random fluctuations displayed some bunching, showing periodic anomalous events



Environmental Context

Matching the measurements in context against known standards of health:

- CO levels occasionally exceeded more than 5 mg/m^3, approaching levels of concern
- NOx levels frequently exceeded more than 100 ppb at rush hour
- Levels of benzene were in normal urban ranges (0.5-10 μg/m^3)

These findings provided useful information for feature engineering and model construction, highlighting the importance of time-based features (hour, day of week) and lag variables in accounting for the temporal correlations in pollutant levels.

Analysis of Potential Factors Influencing Air Quality Variations: There are many factors which affect air quality changes including anthropogenic activities like transportation, industrialization, domestic fossil fuel usage, meteorological factors like temperature, humidity, wind speed, seasonality. The main source of emissions is traffic emissions, particularly diesel vehicles, with more NOx emitted at peak hours. Industrialization releases harmful pollutants like NOx and Benzene through combustion of fossil fuels and chemical reactions, enhanced in urban areas with industrial zones. Domestic use of fossil fuels is responsible for particulate and CO emissions. Meteorological parameters like temperature, humidity, wind speed, and seasonality also play a role in pollutant concentration levels. Urban layout and population density also contribute to air quality variations. Chemical reactions and seasonal variations in pollutants like carbon monoxide, nitrogen oxides, and benzene also lower air quality. These drivers need to be addressed through specific interventions like encouraging cleaner transport modes, managing industrial emissions, enhancing urban planning, and boosting renewable energy use.

Modeling Approach and Results

Feature Engineering

Based on the exploration findings, several feature sets were engineered to improve prediction accuracy:

1. Time-based Features: Day of week, hour of day, month. These features show the variation along the course of a week.

2. Lagged Features: Previous 1, 3, 6, 12, and 24-hour CO concentrations. These capture temporal dependencies identified in autocorrelation analysis

3. Rolling Statistic: 6, 12, and 24-hour rolling means and standard deviations. These capture recent trends and volatility of real-time data.

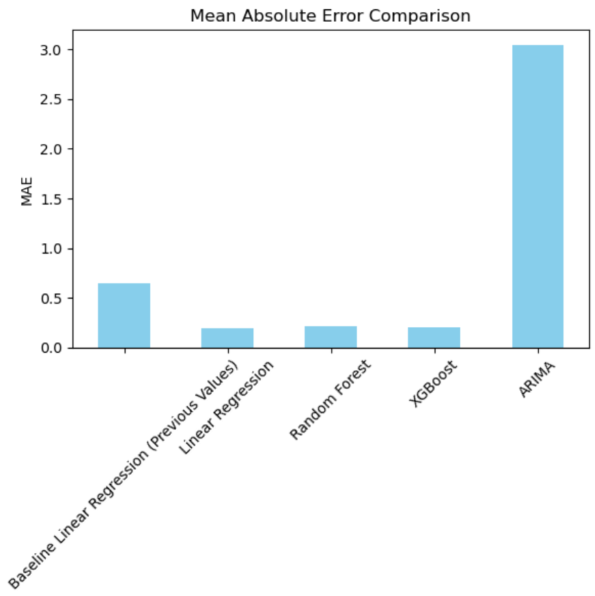
Model Training and Evaluation

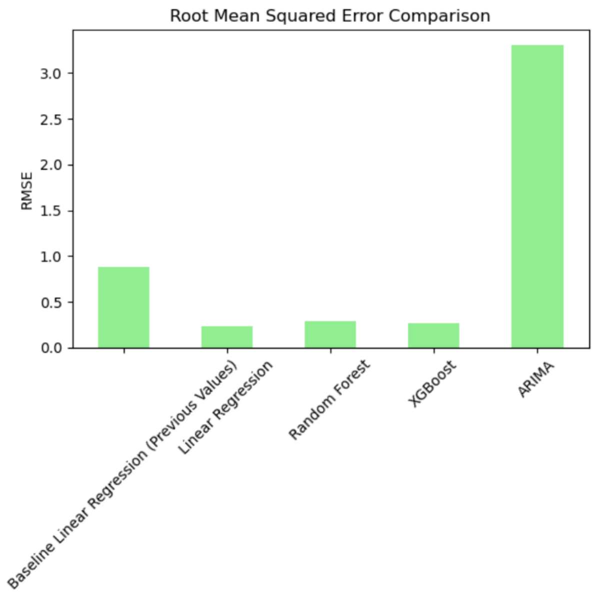
Baseline Linear Regression (Previous Values):
The baseline model used lagged features to provide comparisons with the current point of time models.
Metrics: MAE = 0.649259, RMSE = 0.886833

Four different modeling approaches were implemented and evaluated:

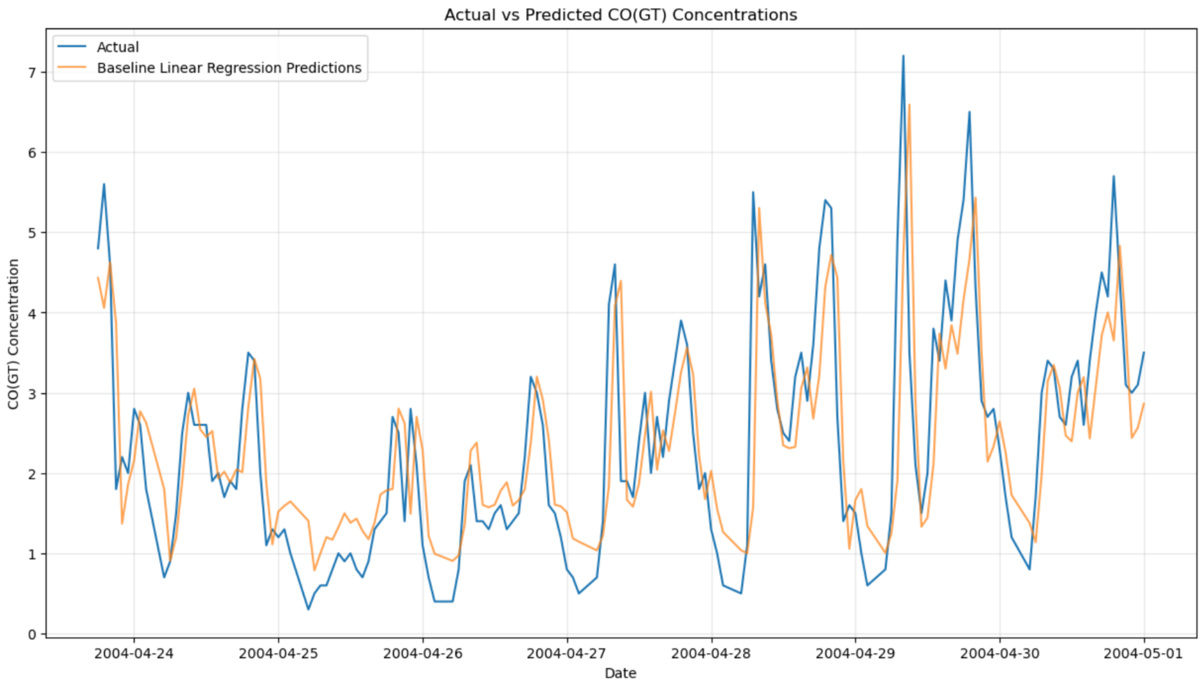
Model	**Description**	**MAE**	**RMSE**
Linear Regression	Implemented model using current point of time	0.191133	0.235631
Random Forest	Leveraged ensemble methods to capture non-linear relationships	0.220031	0.288133
XGBoost	Gradient boosting implementation	0.205189	0.265656
ARIMA	Time series specific model	3.041043	3.305664

The comparison based on the metrics revealed:

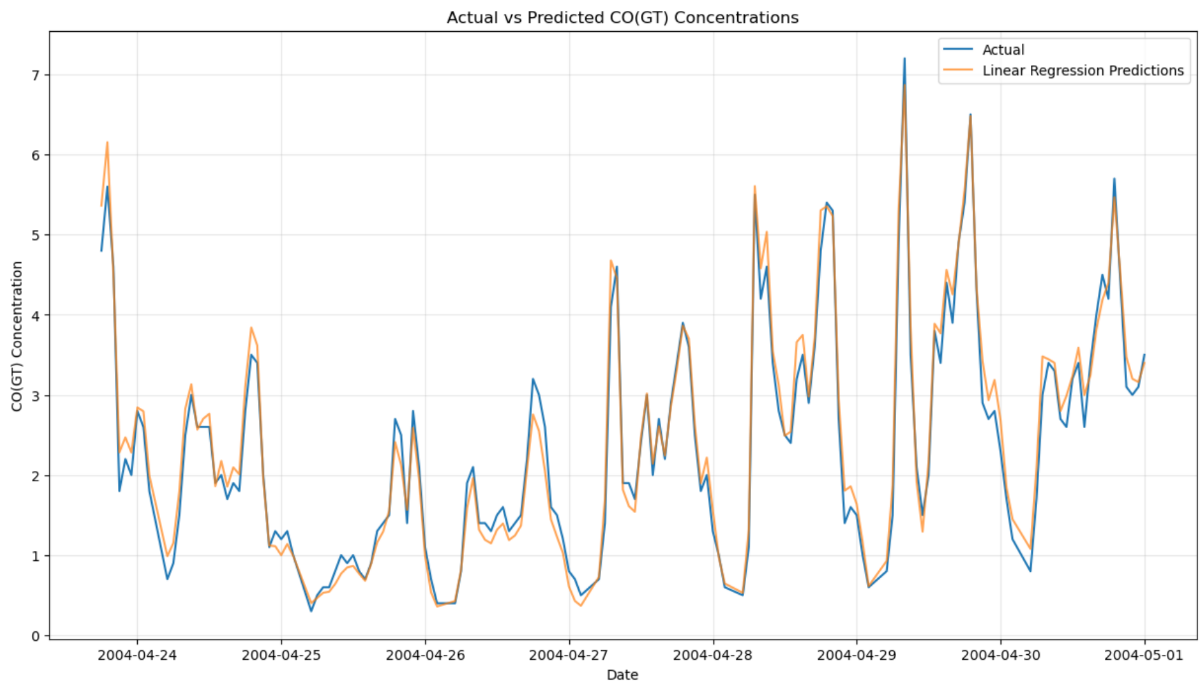




The predicted concentrations using lagged features and their actual readings varied a lot as seen below.



Among all the models, the Linear Regression model was better than more complex models. The actual data versus the predicted data by linear regression model aligned closely as seen below.



This suggested that:
1. The mapping between features and target is predominantly linear
2. The designed features successfully modeled the underlying patterns
3. More complex models might have overfit to noise

As it performed the best, the linear regression trained model was saved using joblib for later use

Kafka Integration for Real-time Prediction

To implement real-time prediction using Kafka:

<u>Producer Implementation</u>:
1. Split data chronologically (80% training, 20% testing)
2. Modified to stream only the test portion of the dataset
3. Contained appropriate preprocessing to encompass missing values handling
4. Incorporated timing synchronization in order to mimic real-time inputs

<u>Consumer Implementation</u>:
1. Loaded the trained model
2. Processed the incoming real-time messages
3. Applied the model to make predictions
4. Stored results in predictions.csv

Prediction Results

Analysis of the predictions made on the consumer side revealed:
1. Linear Regression did not differ as much during real-time prediction
2. Mean Absolute Error varied around 0.19, in line with validation performances
3. Prediction closely followed actuals, both trends and day-of-week patterns

A prominent finding was that lag features played an important role in precise prediction. Recent readings led the way in the model, with the largest contribution of the 1-hour lag taking precedence over the 3-hour lag and the 6-hour lag.

Conclusion and Limitations

This project was able to successfully illustrate the application of Apache Kafka with predictive modeling for real-time air quality monitoring. Some of the highlights include:

1. Constructing an effective Kafka-based streaming pipeline for air quality data
2. Uncovering significant temporal trends in pollutant levels
3. Developing effective predictive models with surprisingly good linear performance
4. Facilitating real-time prediction using Kafka

There are, however, some limitations that need to be mentioned:

1. Data Limitations:
   - The dataset covers a short time period (March-April 2004)
   - Limited to a single monitoring station, reducing spatial understanding

2. Model Limitations:
   - Models don't account for extreme weather conditions or pollution incidents
   - Feature set is small and doesn't include potentially important factors like traffic volume data

3. System Limitations:
   - Current implementation doesn't provide backpressure for high-volume scenarios
   - No support for retraining models as new data arrives

Possible future improvements would include:

1. Implementing a sliding window mechanism for continuous model retraining
2. Integrating multiple data sources (weather forecasts, traffic data)
3. Creating a feedback loop for model drift detection
4. Scaling to distributed deployment across multiple monitoring stations

Overall, this project demonstrates the potential of combining streaming technologies and predictive modeling for environmental monitoring. The approach provides a foundation for more sophisticated real-time monitoring systems that can enhance public health response and urban planning decisions.