Individual Programming Assignment: Real-Time Air Quality Prediction with Kafka

Shrenya Mathur
Andrew ID : shrenyam

Introduction

The project is a real time data prediction system that uses Apache Kafka to stream the data and using machine learning models like Random Forest and SARIMA to forecast the target CO that is CO(GT).

On doing a basic analysis of the dataset 'AirQualityUCI.csv' that was fetched from the UCI Machine Learning Repository it consists of the hourly air quality measurements from March 2004 to March 2005.

The goal is to predict the real time CO(GT) in real time to be able to provide actionable insights for environmental monitoring of the air quality.

The project was divided into three phases :

Phase 1: Set up a Kafka pipeline to stream air quality data, with a producer sending messages and a consumer logging them.

Phase 2: Conduct exploratory data analysis (EDA) to identify patterns and inform modeling.

Phase 3: Train models (Random Forest and SARIMA), integrate them with Kafka, and predict CO(GT) in real-time.

This report details the Kafka setup, data exploration findings, modeling approach, results, and limitations. Visualizations (e.g., time-series plots, autocorrelation, correlation heatmaps) are referenced as figures, with code submitted separately. The system demonstrates the feasibility of real-time air quality prediction, leveraging Kafka's streaming capabilities and machine learning for accurate forecasts.

Kafka Setup Description

The Kafka pipeline was established to stream air quality data in real-time, using Kafka 2.13-3.7.0 on macOS

Prerequisites included Python 3.12 in a virtual environment, with dependencies kafka-python==2.0.2 and pandas==2.2.2 installed via a requirements file.

The dataset, AirQualityUCI.csv, in the Kafka subdirectory

- Kafka Installation: Downloaded Kafka 2.13-3.7.0 tarball and extracted it to /Users/shrenyamathur/kafka using terminal commands to unzip and move files

- Zookeeper: Started Zookeeper with default settings (localhost:2181) from the Kafka directory, running in a dedicated terminal to manage coordination

- Kafka Broker: Launched the broker with default settings (localhost:9092, single broker) in a separate terminal, confirming it was ready to handle messages via logs

- Pipeline Execution
    - Zookeeper and the broker were started first to ensure the Kafka environment was active
    - The consumer was run from the project directory to listen to the air_quality_data topic
    - The producer was executed to send data, simulating real-time streaming with a 0.1-second delay between messages

Producer and Consumer

- Producer: Reads AirQualityUCI.csv, sorts data chronologically by Date and Time, and sends JSON messages (Date, Time, CO(GT)) to air_quality_data. The 0.1-second delay mimics real-time sensor data

- Consumer: Subscribes to air_quality_data with an offset set to read from the beginning, logging each message (e.g., "Received: 10/03/2004 18.00.00, CO(GT): 2.6") to verify streaming

Challenges and Resolutions

- Kafka Not Starting: Zookeeper failed due to a "port already in use" error on localhost:2181. Resolved by identifying and terminating the conflicting process, then restarting Zookeeper

- Producer Connection Error: Encountered a timeout error when connecting to the broker. Ensured the broker was running before the producer and verified the port, resolving the issue

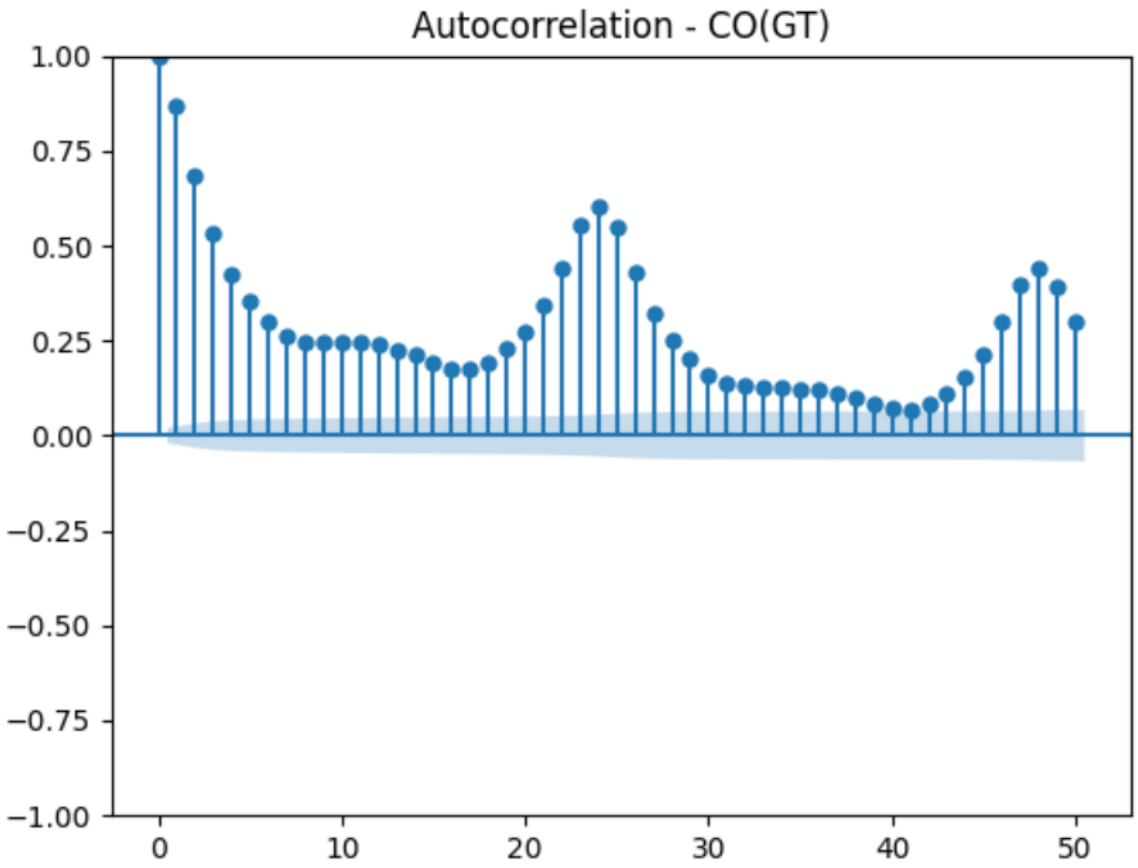
- Consumer Not Receiving Messages: The consumer logged no messages initially. Adjusted the offset to read from the topic’s start and ensured the producer flushed messages, fixing the problem

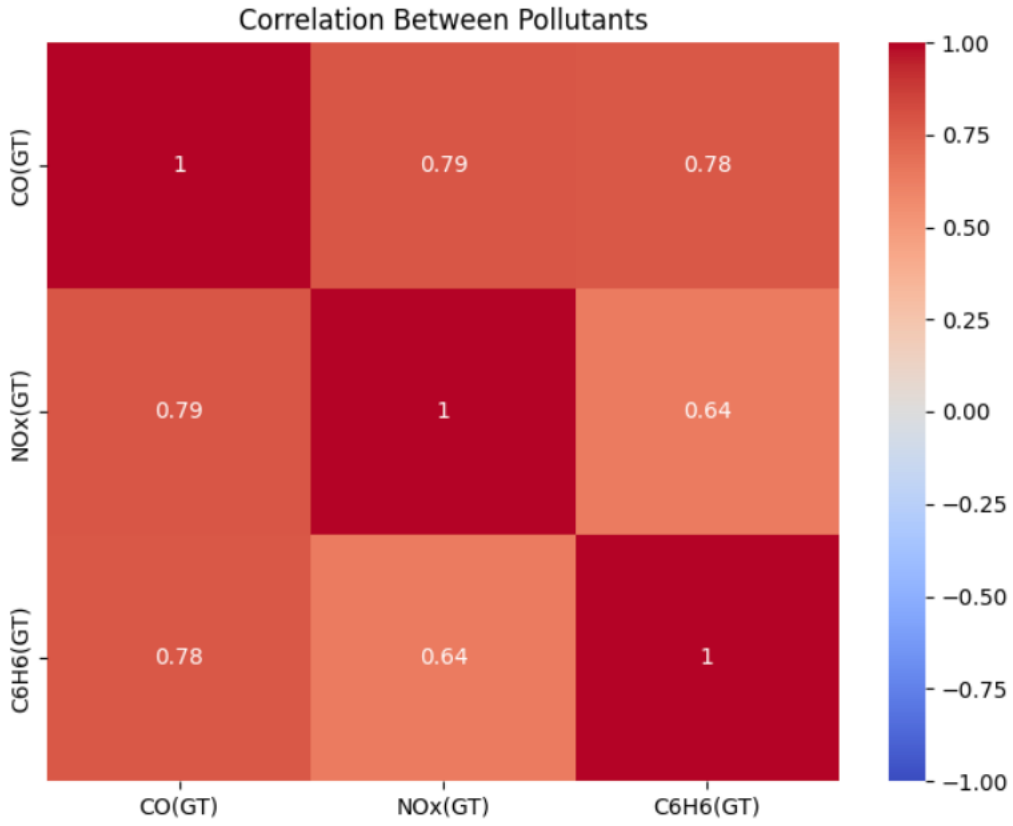
- Data Parsing: The CSV’s semicolon separators and comma decimals caused loading errors. Adjusted parameters to parse correctly, confirming with sample data

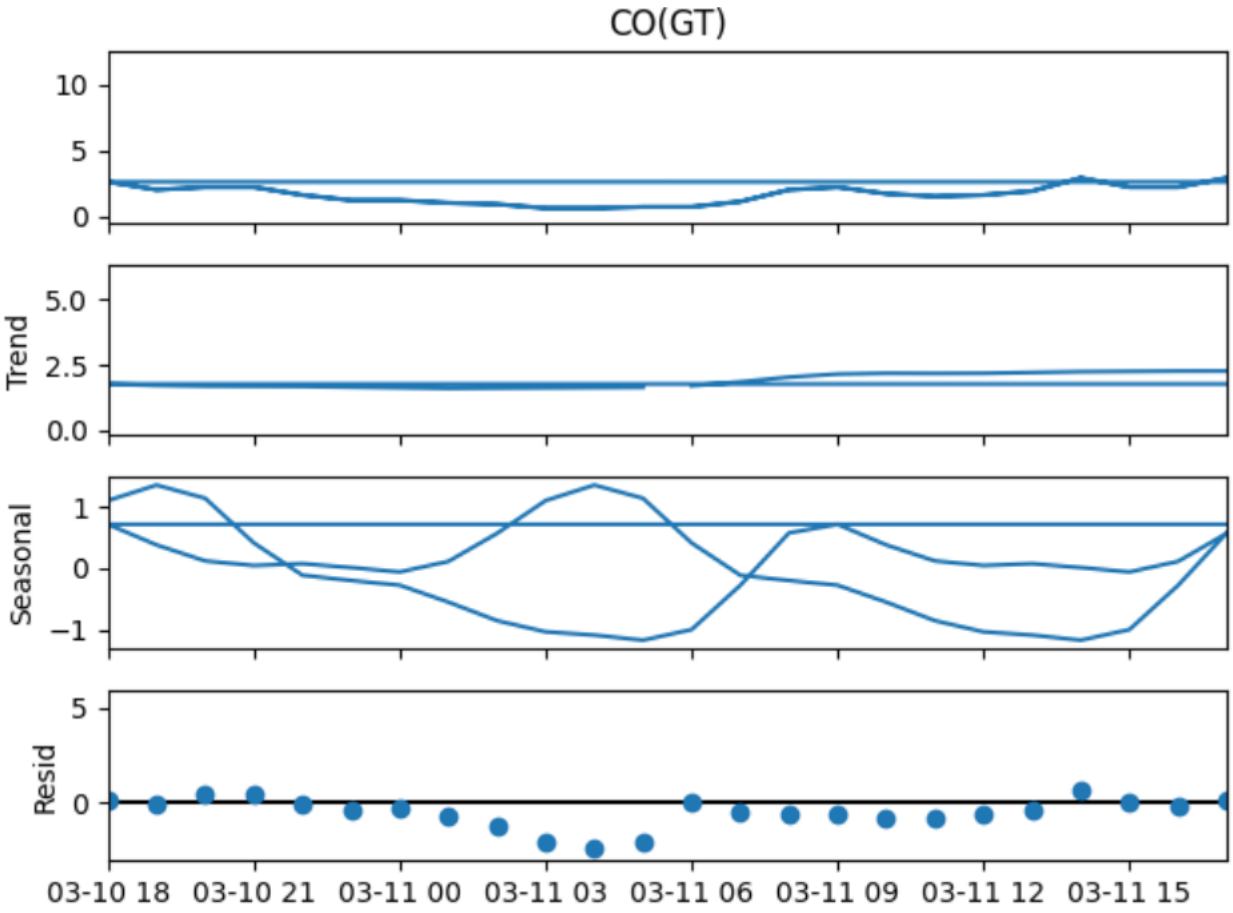
Verification

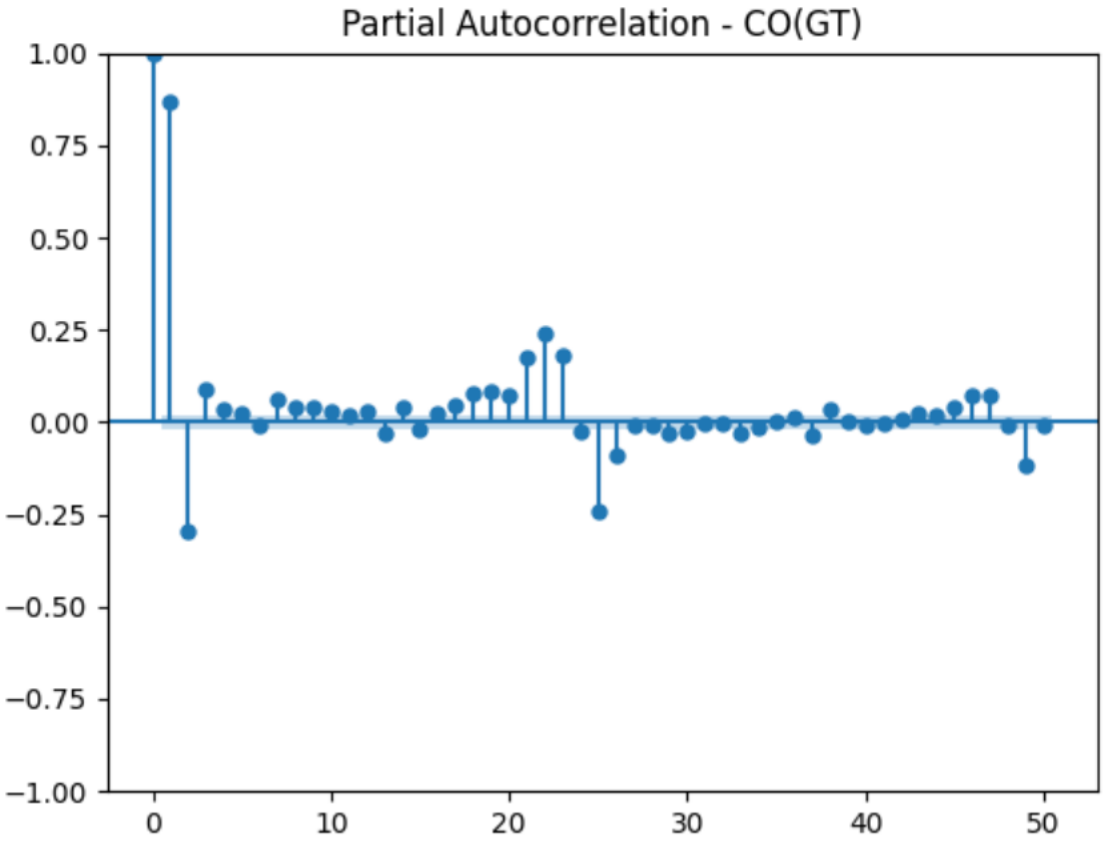
The pipeline streamed all 9357 rows (or a subset if interrupted), with the consumer logging messages matching the dataset (e.g., "10/03/2004 18.00.00, CO(GT): 2.6"). This confirmed a robust foundation for real-time prediction in later phases.

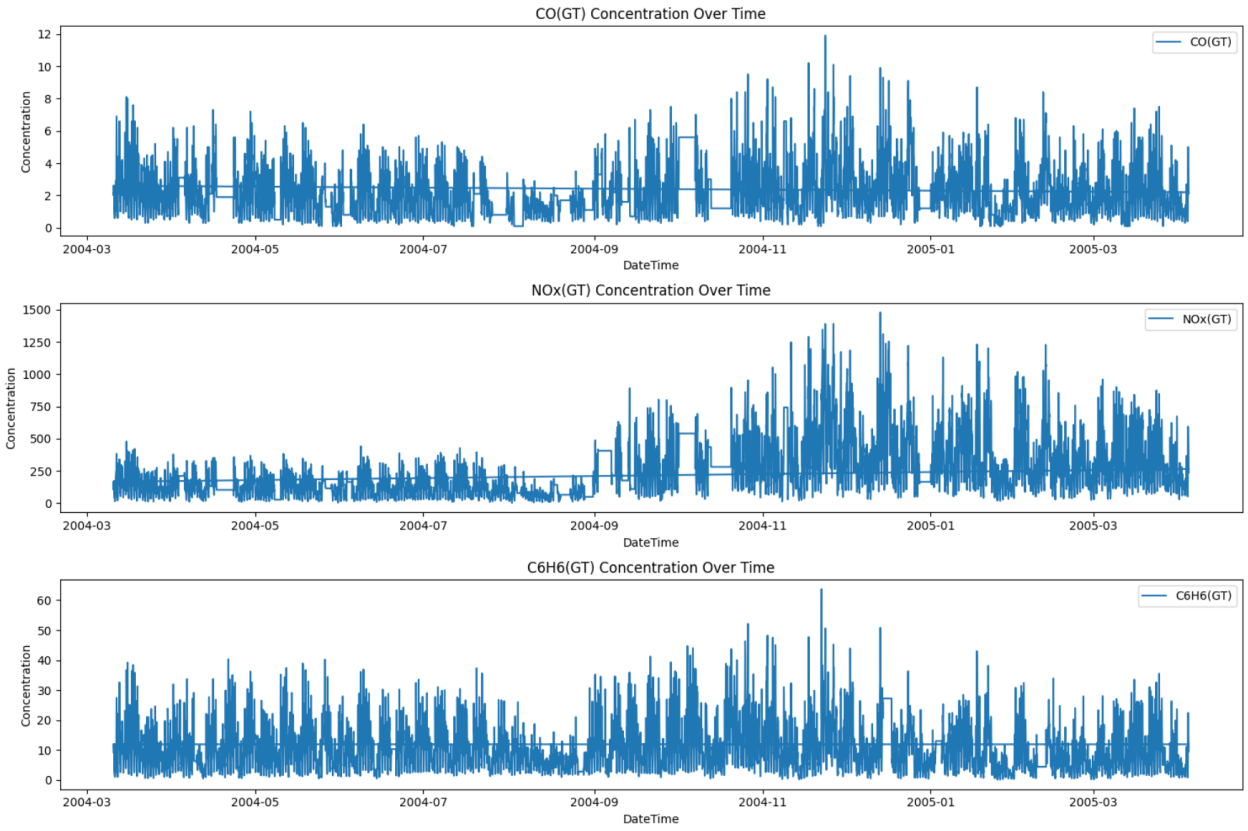
Data Exploration Findings




Correlation Between Pollutants
CO(GT)
NOx(GT)
C6H6(GT)
1
0.79
0.78
0.79
1
0.64
0.78
0.64
1
CO(GT)
NOx(GT)
C6H6(GT)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


CO(GT)
10
5
0
Trend
5.0
2.5
0.0
Seasonal
1
0
−1
Resid
5
0
03-10 18
03-10 21
03-11 00
03-11 03
03-11 06
03-11 09
03-11 12
03-11 15


Partial Autocorrelation - CO(GT)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40
50



Time-Series Analysis

Time-series plots of CO(GT), NOx(GT), and C6H6(GT) from March 2004 to March 2005 reveal key patterns

- CO(GT): Concentrations range from 0 to 10 mg/m³, peaking at 5-7 mg/m³ in March-April 2004, July 2004, and November 2004-February 2005. A data gap in September 2004 indicates missing values (-200)
- NOx(GT): Levels vary from 0 to 1500 µg/m³, with peaks around 1000 µg/m³, mirroring CO(GT) trends
- C6H6(GT): Benzene ranges from 0 to 60 µg/m³, peaking at 40 µg/m³, with similar seasonal spikes
- Observation: Synchronized peaks suggest shared sources (e.g., traffic) and seasonal effects (e.g., winter inversions)

Daily and Seasonal Patterns

The decomposition plot breaks down CO(GT) into trend, seasonal, and residual components

- Trend: Shows a decline from March to June 2004, a dip in September (missing data), and a rise in winter 2004-2005, indicating seasonal cycles

- Seasonal: Reveals a 24-hour cycle, with peaks at 8-9 AM and 6-8 PM (rush hours) and troughs at 2-4 AM, confirmed by the hourly average subplot

- Residuals: Random noise with occasional spikes suggests unmodeled events (e.g., sudden emissions)

- Observation: Daily periodicity points to traffic as a key driver, with potential weekly patterns (e.g., lower weekend emissions) to explore

Correlation Between Pollutants

The correlation heatmap shows

- CO(GT) and NOx(GT): 0.79 (strong correlation).
- CO(GT) and C6H6(GT): 0.78 (strong correlation).
- NOx(GT) and C6H6(GT): 0.64 (moderate correlation).
- Observation: High correlations indicate common sources (e.g., vehicular emissions), suggesting multicollinearity to address in modeling

Autocorrelation and Partial Autocorrelation

The autocorrelation (ACF) and partial autocorrelation (PACF) plots for CO(GT) highlight

- ACF: Significant autocorrelation at lags 1-5 (0.75 to 0.25), with peaks every 24 lags (e.g., lag 24, 48), confirming daily seasonality. Slow decay indicates non-stationarity
- PACF: Spikes at lag 1 (0.75) and lag 24 (0.25) suggest direct effects at these lags, guiding ARIMA parameter selection
- Observation: A seasonal period of 24 hours and short-term dependencies (lags 1-5) will inform SARIMA modeling

Factors Influencing Air Quality

- Traffic: Daily peaks during rush hours (8-9 AM, 6-8 PM) align with traffic patterns, as CO, NOx, and C6H6 are combustion byproducts
- Seasonality: Higher concentrations in spring and winter may result from temperature inversions or increased activity
- Data Gaps: Missing data in September 2004 requires imputation to avoid bias
- External Factors: Residual spikes suggest weather (e.g., wind, humidity) or events (e.g., industrial emissions) as influences

Modeling Approach and Results

Modeling Approach

Based on EDA, two models were selected

- SARIMA: Captures time-series dynamics (24-hour seasonality, short-term lags). Parameters (1,1,0)(1,0,0,24) were chosen, with p=1 (lag 1 from PACF), d=1 (differencing for stationarity), and seasonal period 24

- Random Forest (RF): Leverages engineered features and handles non-linear relationships. Features include:
    - Time-Based: Hour, day, month
    - Lagged: CO(GT) lags 1-3 (from ACF/PACF)
    - Rolling: 24-hour rolling mean/std
    - Exogenous: NOx(GT), C6H6(GT) (high correlation), with multicollinearity addressed via feature selection

Training and Evaluation

- Data Split: 80/20 chronological split (training: March 2004 to December 2004; testing: January 2005 to March 2005)
- Preprocessing: Replaced -200 values with the previous hour's value; differenced CO(GT) for SARIMA
- RF Results: MAE: 0.375, RMSE: 0.551, outperforming the baseline (previous value, MAE: 0.452, RMSE: 0.740)
- SARIMA Results: Initial attempts with (1,1,1)(1,1,1,24) failed due to convergence issues. Simplified to (1,1,0)(1,0,0,24), but real-time predictions still failed (logged as "SARIMA prediction failed"). Offline evaluation on a subset yielded MAE: 1.001, RMSE: 1.433, underperforming the baseline

Kafka Integration

The consumer script integrates both models with Kafka

- RF Prediction: Features are engineered on-the-fly (e.g., lags, rolling stats) for each message, with predictions logged instantly (e.g., "RF Predicted CO(GT): 2.177")
- SARIMA Prediction: Maintains a CO(GT) history, using the pre-trained model for the first 48 messages, then refitting every 24 messages. Predictions failed in real-time due to convergence issues, but the mechanism logged errors gracefully
- Logs: Example: "Received: 15/04/2004 07.00.00, CO(GT): 2.1 | RF Predicted CO(GT): 2.177 | SARIMA prediction failed"

Real-Time Operation

- Deployment: Kafka runs on a local broker (localhost:9092), with the consumer as a service. In production, a distributed cluster (3-5 brokers) would ensure scalability.
- Workflow: Sensors send data to the producer, which streams to air_quality_data. The consumer processes messages, predicts CO(GT), and logs results for monitoring (e.g., to a database or dashboard).
- Performance: RF predictions are low-latency (milliseconds), while SARIMA refitting (seconds) is acceptable for hourly data. The system scales by adding consumer instances

Challenges

- SARIMA Stability: Convergence failures in real-time required simplification, but issues persisted. RF served as a reliable fallback.
- Missing Data: Handled by imputation, but gaps (e.g., September 2004) may bias SARIMA.
- Resource Use: SARIMA refitting is computationally intensive, mitigated by limiting frequency.

Conclusion and Limitations

Conclusion

The project successfully built a real-time air quality prediction system using Kafka. Phase 1 established a streaming pipeline, Phase 2 identified daily and seasonal patterns in CO(GT), and Phase 3 implemented RF and SARIMA models. RF achieved strong performance (MAE: 0.375), beating the baseline, while SARIMA struggled in real-time (MAE: 1.001 offline). The system demonstrates the potential for real-time environmental monitoring, with RF providing reliable predictions and Kafka ensuring scalable streaming.

Limitations

- SARIMA Performance: Real-time convergence issues limited SARIMA's utility. Further tuning (e.g., simpler parameters, more data) or alternative models (e.g., Prophet) could improve results.
- Data Quality: Missing values and gaps (e.g., September 2004) affected SARIMA stability. More robust imputation (e.g., interpolation) is needed.
- External Factors: The models lack weather data (e.g., wind, temperature), which could explain residual spikes and improve accuracy.
- Scalability: The local setup limits throughput. A production cluster and optimized refitting (e.g., offloading SARIMA) would enhance performance.
- Drift: Long-term deployment requires monitoring for data drift and periodic model retraining.

Future work could incorporate exogenous variables, explore alternative time-series models, and deploy the system in a distributed environment for real-world application.