**Introduction**

With the rapid pace of urbanization, tracking and forecasting air quality has become vital for safeguarding public health and guiding urban development. Elevated levels of pollutants such as Carbon Monoxide (CO), Nitrogen Oxides (NOx), and Benzene (C6H6) pose serious risks to respiratory health and general well-being. Analyzing real-time air quality data plays a key role in issuing timely alerts, improving traffic management to lower emissions, and supporting informed policy-making.

This report analyzes Air Quality data, which was gathered from a gas multisensor device deployed on the field in an Italian city from March 2004 to February 2005 . The dataset consists of 9357 records of hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device. Parameters in this dataset include Date, Time, Concentration of CO, Non Metanic HydroCarbons, Benzene, NOx, NO2, Temperature, and Humidity. In this report, there will be an explanation about how to set up Kafka to stream the data, findings from data exploration, model fitting, and the results of the model.

**Kafka setup description**

To enable real-time data streaming, Apache Kafka was set up. The process began with the installation of JAVA JDK 17, followed by setting the JAVA\_HOME environment variable.

After that, I download kafka\_2.13-3.5.1.tgz and extract the zipped file and move the extracted file to C:\kafka as displayed in figure 1.

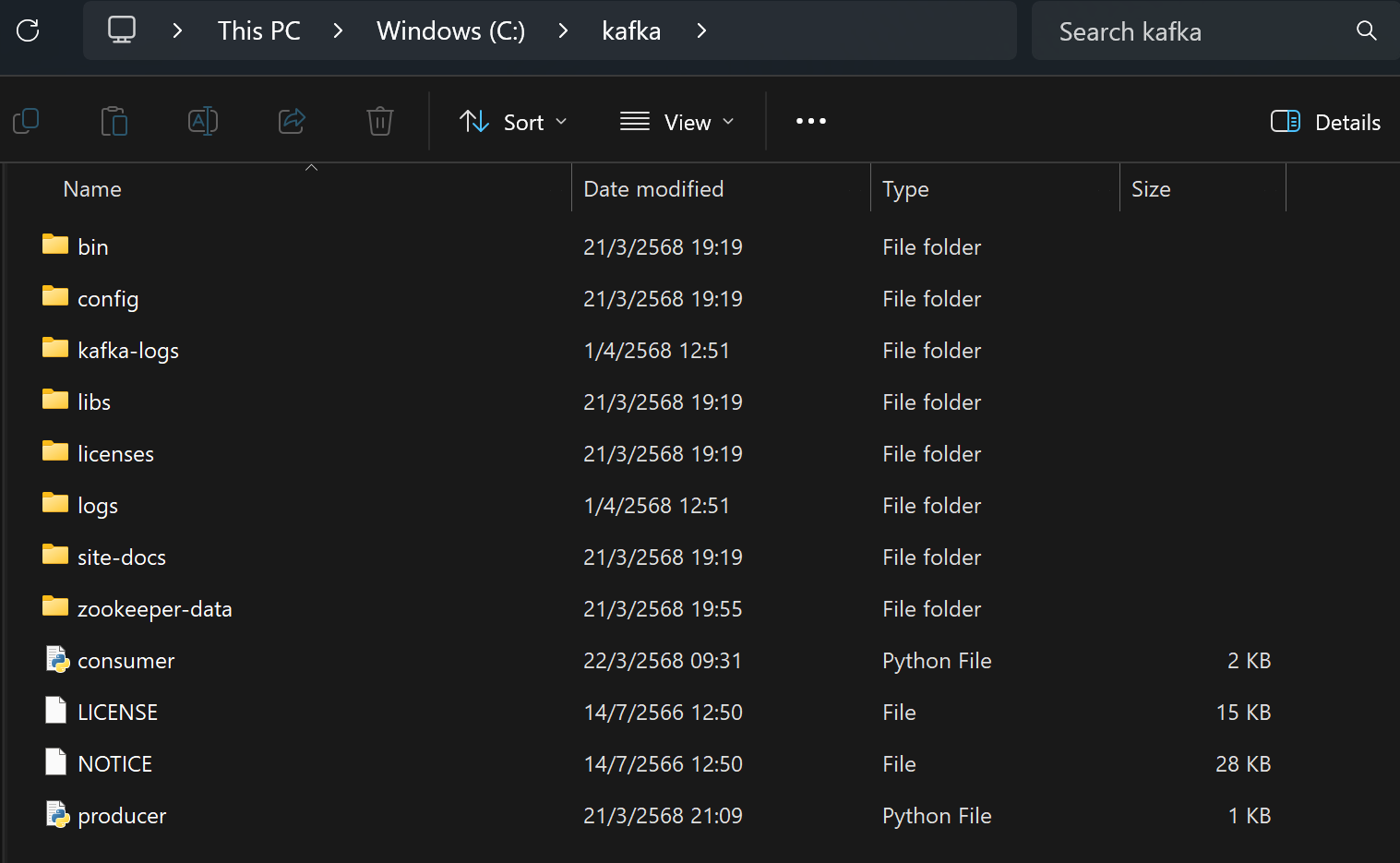


Figure 1: Kafka installation

I then configure ZooKeeper by editing dataDir property in zookeeper.properties file to data Dir=C:/kafka/zookeeper-data and create zookeeper-data directory in kafka folder as shown in figure 1 and figure 2.

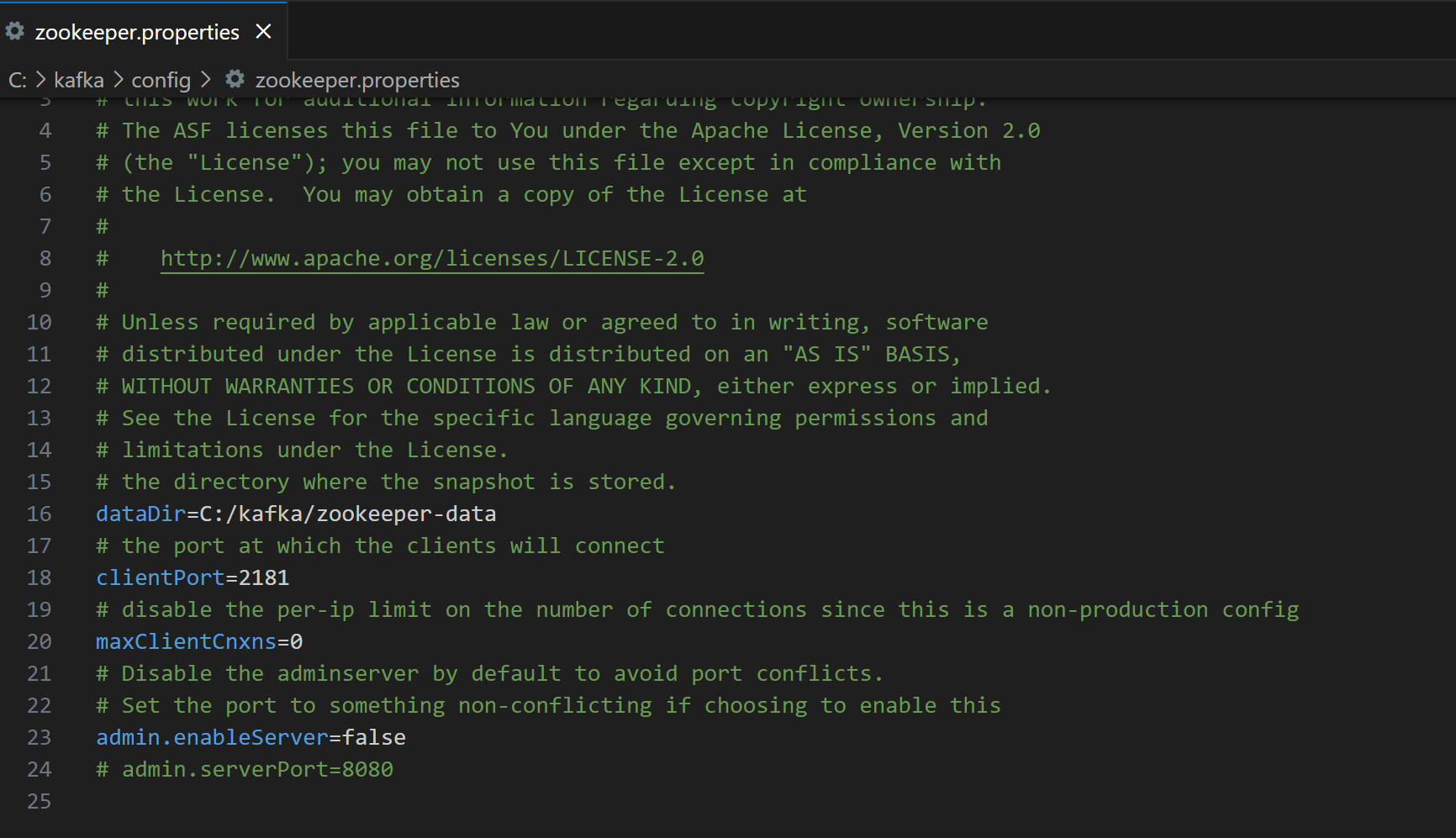


Figure 2: ZooKeeper properties path set up

Next, configure Kafka server by editing log.dirs property in server.properties file to log.dirs=C:/kafka/kafka-logs and create kafka-logs directory in kafka folder as illustrated in figure 1 and figure 3.

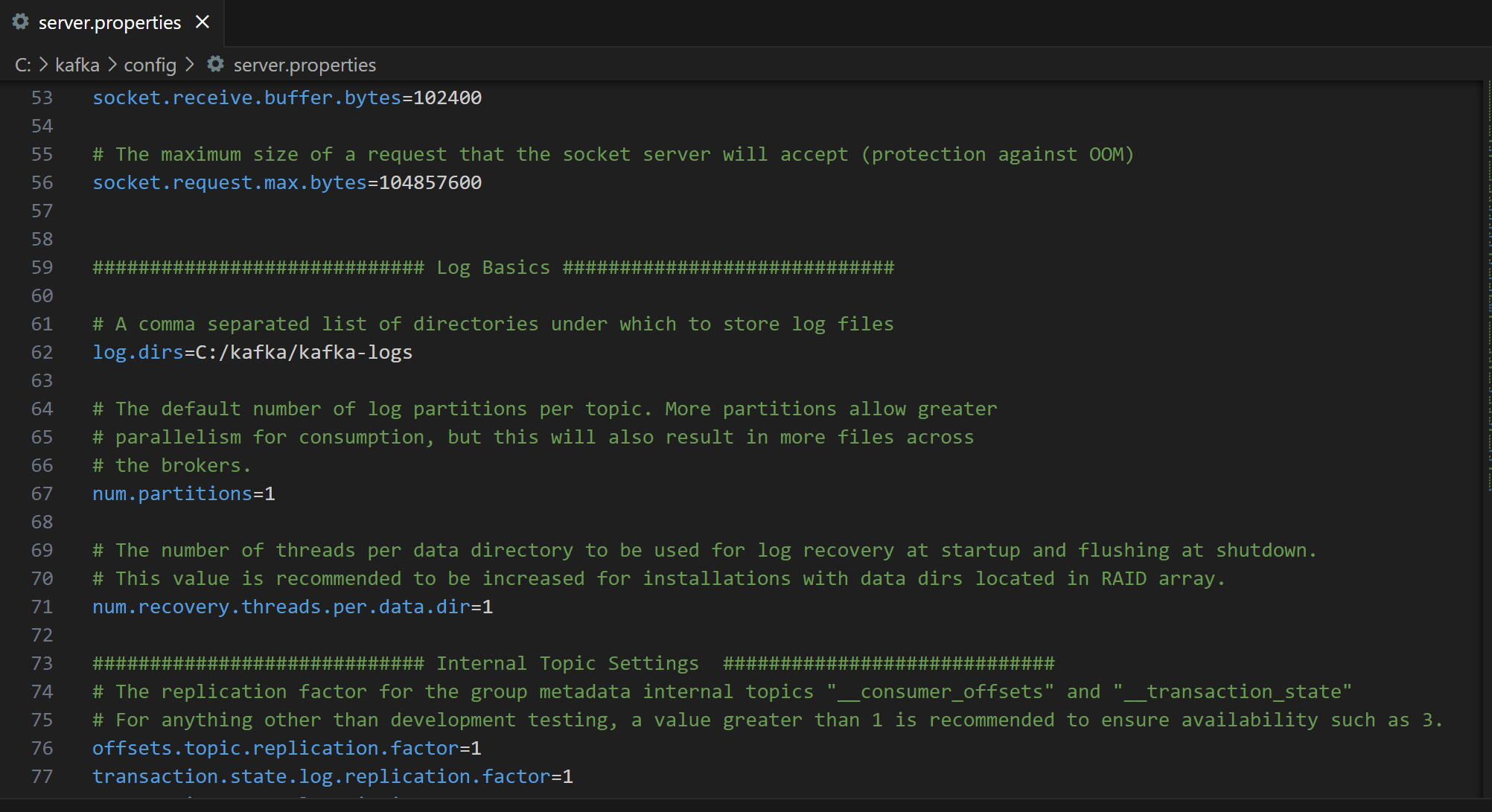


Figure 3: Server properties path set up

Then, I start the ZooKeeper server by using command prompt to navigate to C:\kafka and run the command .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties and get the result as shown in figure 4.

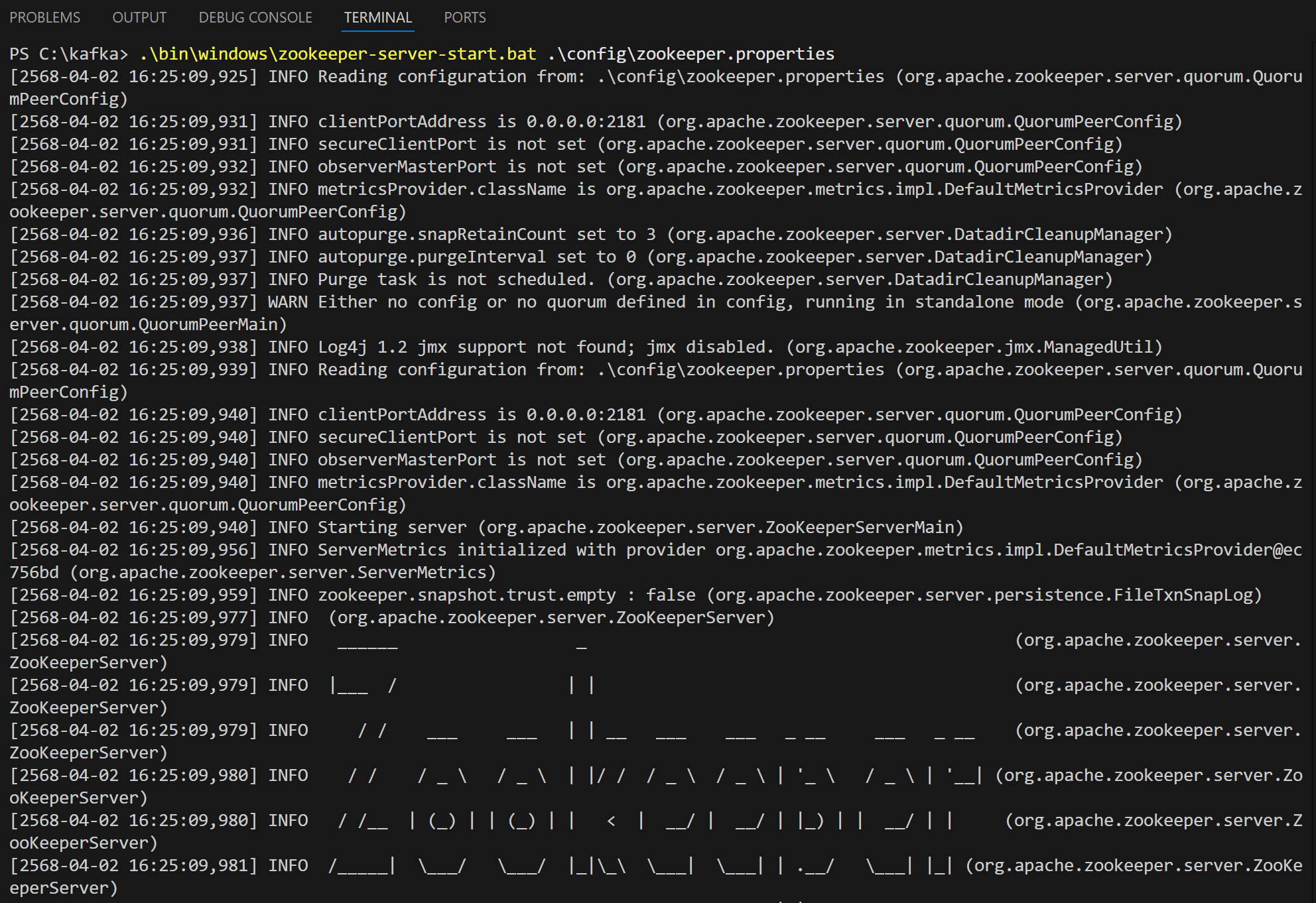


Figure 4: Start ZooKeeper server

After that, I start the Kafka server by using a new command terminal to navigate to C:\kafka and run .\bin\windows\kafka-server-start.bat .\config\server.properties. The result is shown in figure 5.

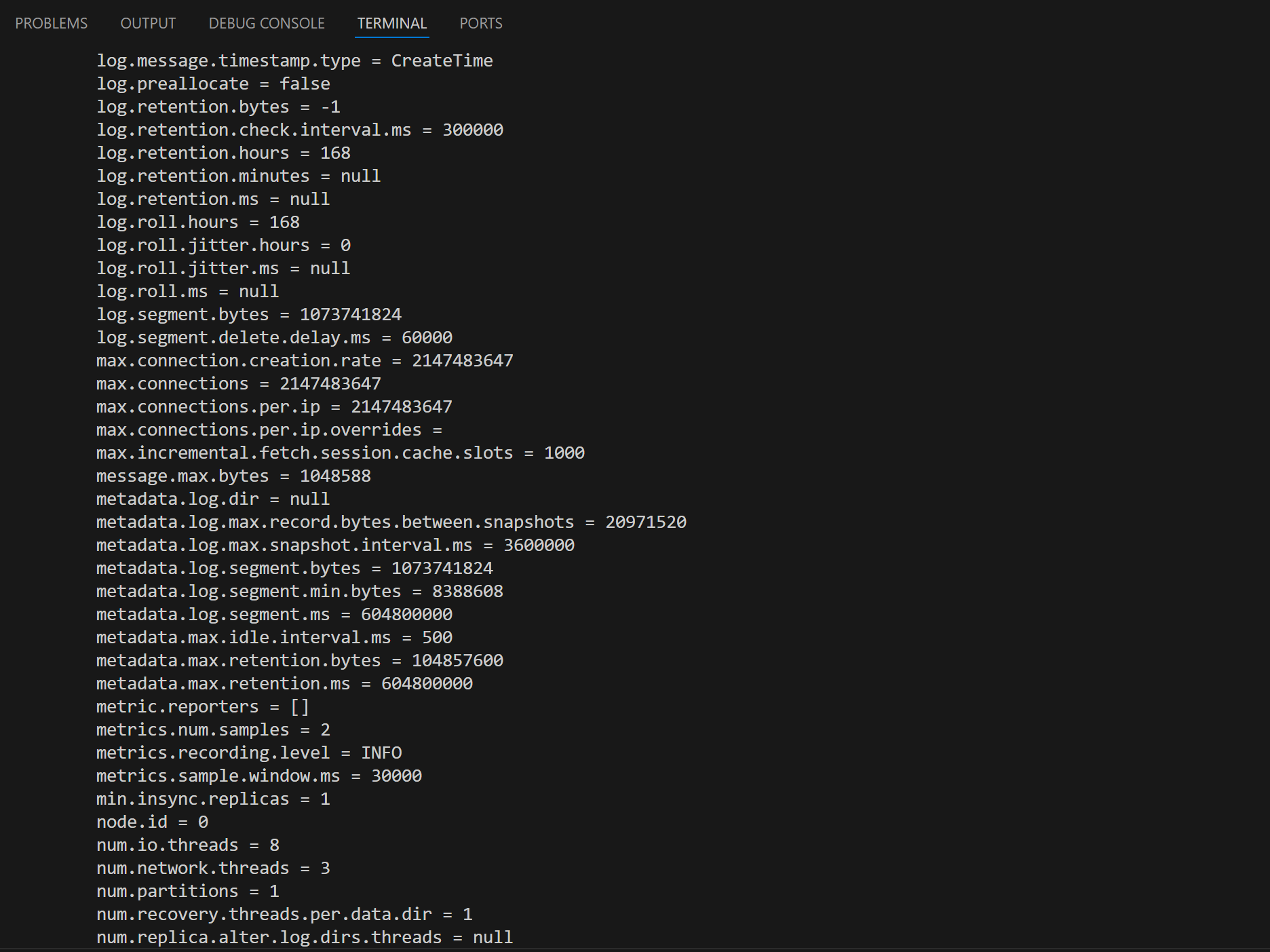


Figure 5: Start Kafka server

After starting the Kafka, the producer is implemented in Python to read the UCI Air Quality dataset and stream only the test dataset to the Kafka topic. Then, the consumer, also implemented in Python, consumes the data from that topic, stores it in a dataframe, as well as predicts the concentrations of CO, NOx, and Benzene from the data it consumes. The consumer consumes data by batch-streaming. It gets data in a batch and makes a prediction for that batch after consuming all the data points.

**Data exploration findings**

Analyzing the time-series plots of CO, NOx, and Benzene concentrations, I found that these three chemicals peak around the end of year 2004. CO and Benzene concentration level maintains the same level with fluctuations and has the lowest value around July - August, while NOx concentration starts from the low level in March 2004 and the value grows during September 2004 and maintains the new level until March 2005. The visualizations are shown in figure 6, 7, and 8.

| Figure 6: CO concentration level | Figure 7: NOx concentration level |
| --- | --- |
| Figure 8: Benzene concentration level | Figure 9: CO concentration level daily patterns |

By analyzing the daily patterns, which average concentration level by hour of day, it is noticeable that CO, NOx, and Benzene concentration levels have similar patterns. They both peak at 9 AM and 7 PM as illustrated in figure 9, 10, and 11.

| Figure 10: NOx concentration level daily patterns | Figure 11: Benzene concentration level daily patterns |
| --- | --- |
| Figure 12: CO concentration level weekly patterns | Figure 13: NOx concentration level weekly patterns |
| Figure 14: Benzene concentration level weekly patterns | Figure 15: Correlation heatmap of pollutants |

As displayed in figure 12, 13, and 14, the weekly patterns of CO, NOx, and Benzene concentration levels also have similar patterns. They increase from Monday to the peak on Friday, and significantly drop on Saturday through Sunday.By analyzing the correlation between each pollutant, the highly correlated ones are CO and Benzene, NOx and NO2, CO and NOx respectively. Figure 15 depicts the correlation matrix of every pollutant.

From the result of Autocorrelation, the plot shows high lag dependency and seasonality of CO concentrations level. The 90 points plot shows seasonality occurs around every 30 days. The same is true for NOx and Benzene. The patterns of these 3 pollutants are similar. They all have significant lag as the bar exceeds the blue areas, which are the significance bounds. The Partial Autocorrelation plots show that there are spikes in the first and second bar as well as during each seasonality effect. These patterns indicate these lags in concentration of these 3 pollutants are important in modeling the data. The patterns are depicted in figure 16, 17, and 18.

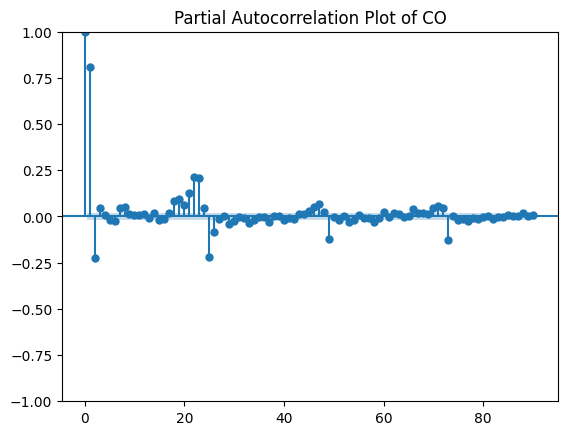
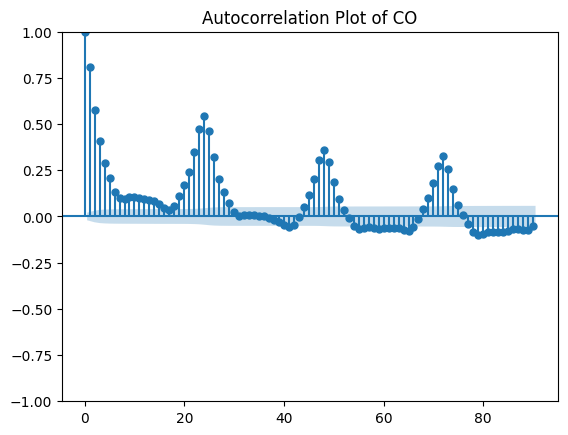


Figure 16: Autocorrelation and Partial Autocorrelation plot of CO concentrations

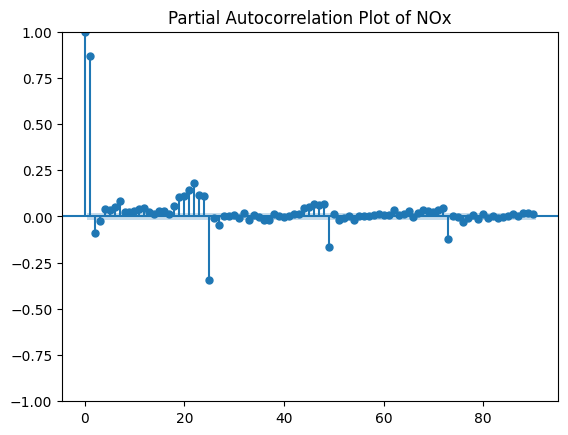
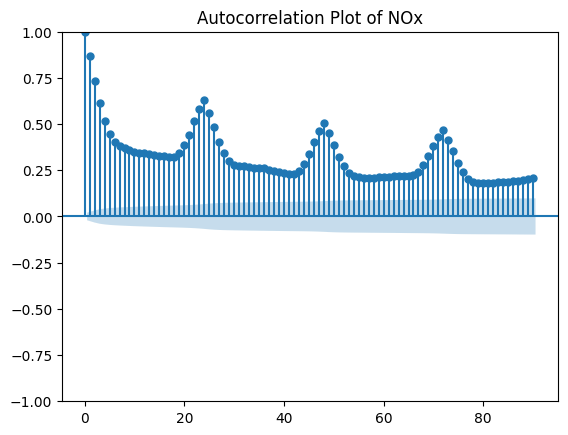


Figure 17: Autocorrelation and Partial Autocorrelation plot of NOx concentrations

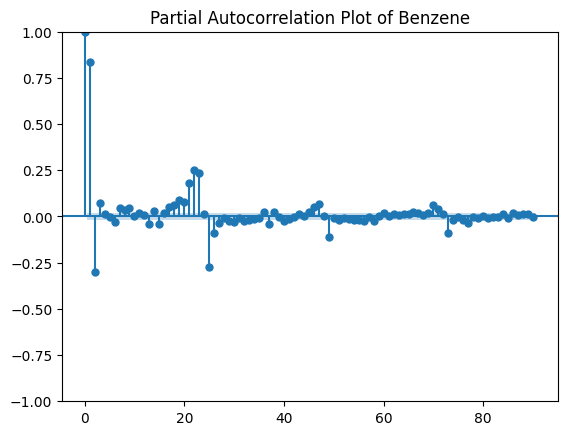
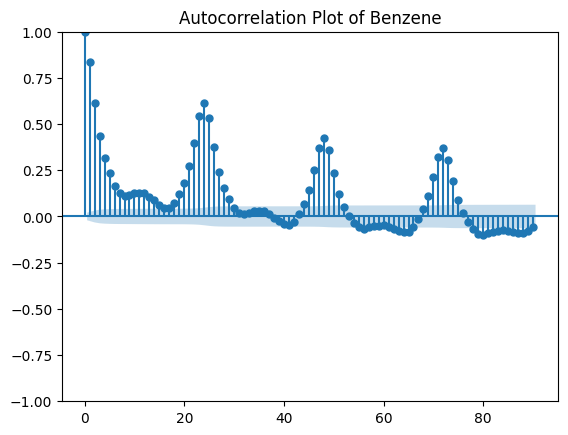


Figure 18: Autocorrelation plot and Partial Autocorrelation plot of Benzene concentrations

When analyzing concentrations of CO, NOx, and Benzene by decomposing the time series into trend, seasonality, and residuals, the results show a quite steady trend during the first couple of months. The concentrations of these chemicals then drop to the lowest after that trend. After that there is an increase trend around 1-2 months and the concentrations decline with some fluctuations in the last few months. It seems like there is a seasonality within approximately 100 data points for all of the concentration levels of these 3 chemicals. However, there are still residuals that cannot be explained by these trends and seasonalities. The illustration can be found in figure 19, 20, and 21.

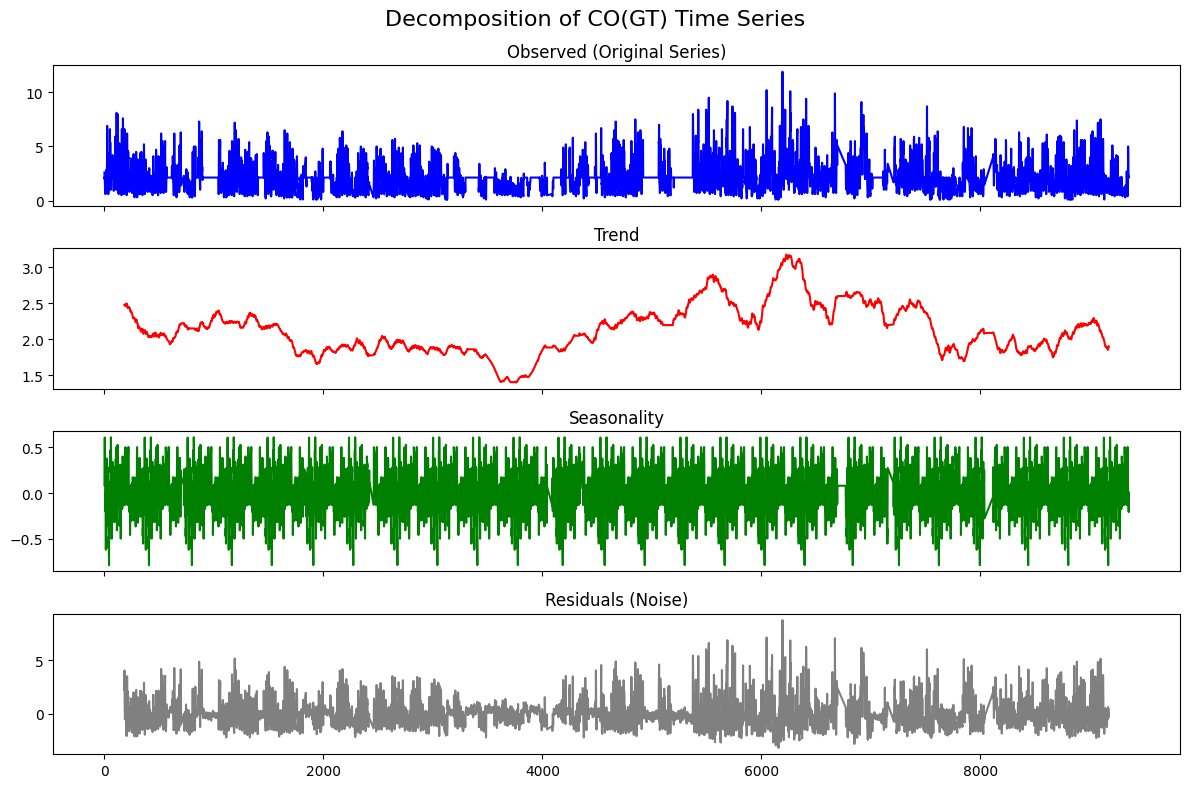


Figure 19: Decomposition of CO(GT) time series

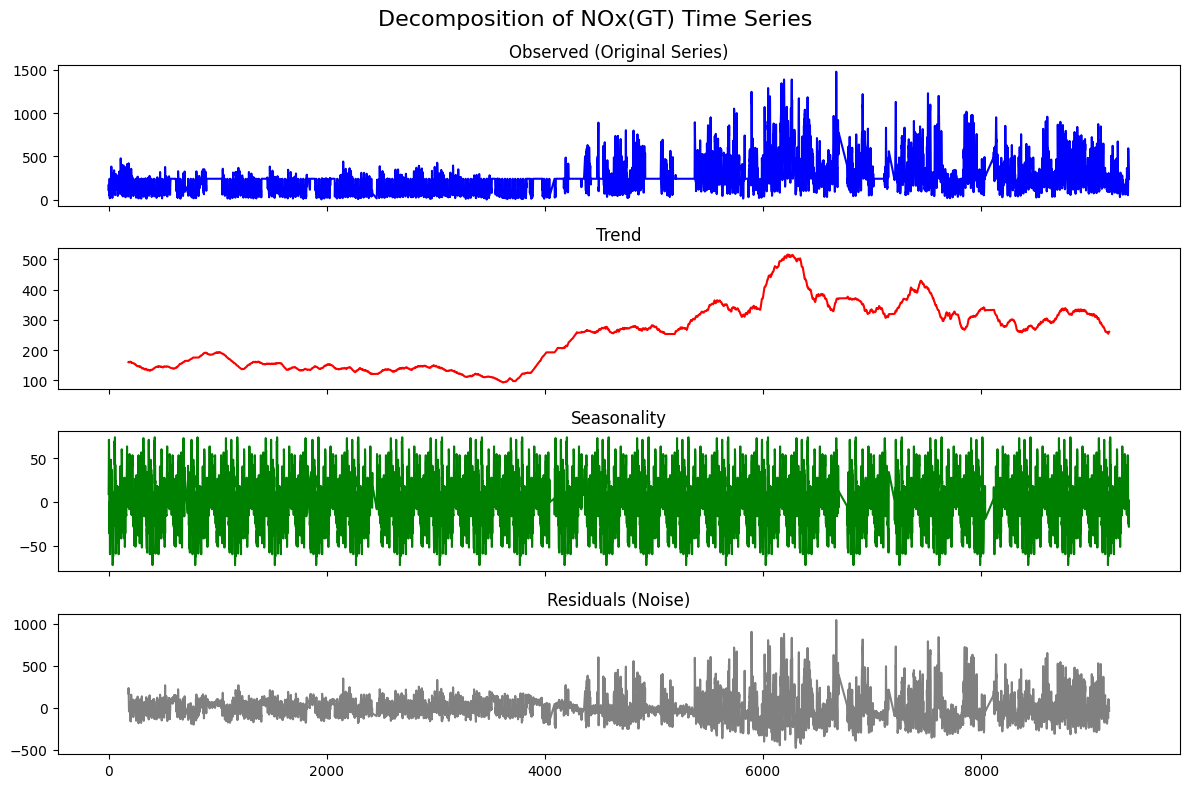


Figure 20: Decomposition of NOx(GT) time series

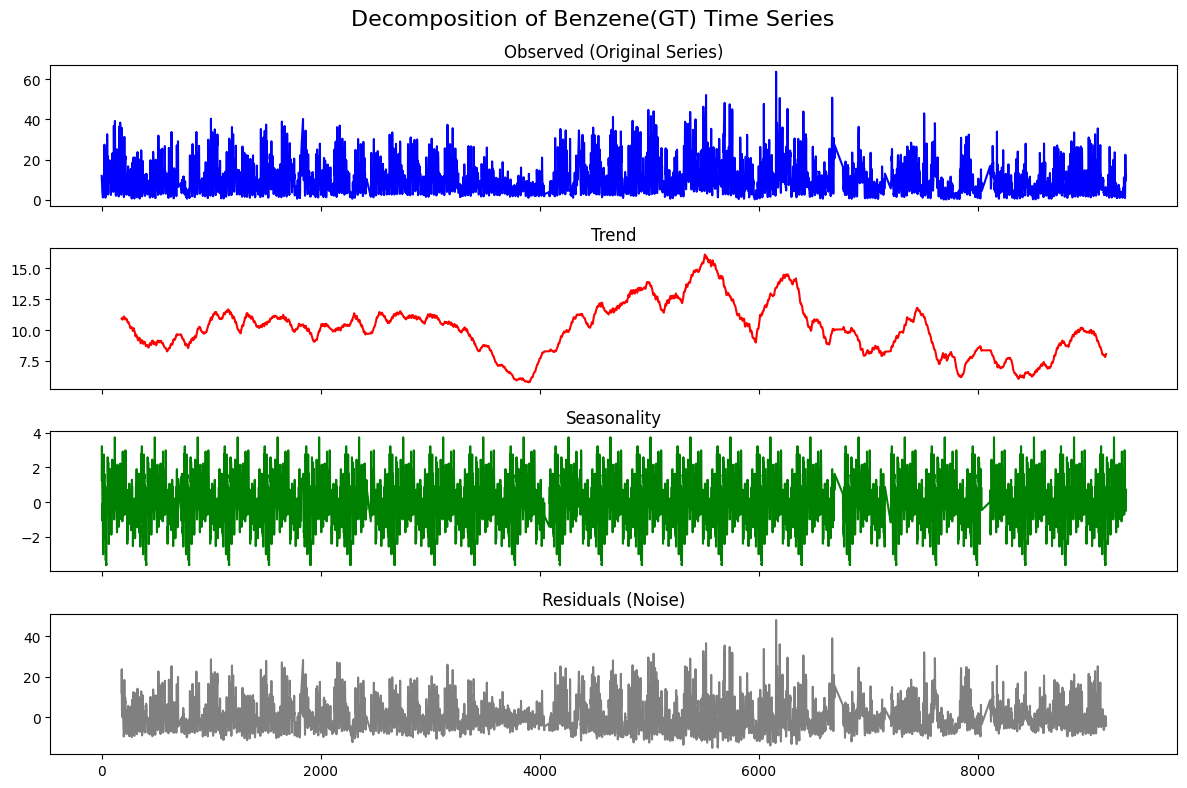


Figure 20: Decomposition of Benzene(GT) time series

In summary, from analyzing daily patterns and weekly patterns, I found that hour in a day and day of the week have effects on the air quality variations. The analysis of correlation between each pollutant indicates that the level of one pollutant has an effect on the others. The Autocorrelation plot and Partial autocorrelation plot show the significant effect of the previous period of the time series as well as the seasonality effect.

**Modeling approach and results**

The first step of implementing a model is to clean the data. I preprocess the data before using it to train and test a model. I checked the missing values in terms of NAs and -200. For missing values (-200) of data in this list PT08.S1(CO), C6H6(GT), PT08.S2(NMHC), PT08.S3(NOx), PT08.S4(NO2), PT08.S5(O3), T, RH, and AH, I replace the missing value with representative value, mean since there are many missing value. I think if I drop out the whole record that contains missing values in this list of parameters, I might miss a significant amount of data I can use to train the model. However, for missing CO(GT), NOx(GT), and NO2(GT), because there are not many missing values and the data is highly related to the prediction I want to make, I decided to drop the record that one of these three variables is missing. Lastly, for NMHC(GT), since there are a lot of missing values, this is not relevant to the prediction, and the remaining data is not enough to train the model, I decided to ignore this parameter.

After cleaning the data, I splitted the dataset into 2 subsets, training set (80% of the total data) and test set (20% of the total data).

The next step is feature engineering and model selection. The insights from data visualization contribute to the way to create models for this dataset. As I found out that there is a pattern in the hours of day, I use hour as one of the features. There is also a pattern in day of week, so I use day of week as a feature. The same logic applies to seasonality. As I noticed seasonality in the dataset, I include day of year, day, month, and year as the features to predict concentrations of pollutants. Lastly, from Autocorrelation and Partial autocorrelation, I consider using lagged features from the previous time periods and average statistics such as mean and standard deviation to capture the lagged nature of this dataset. Leveraging the insights from correlation heatmap, I use NOx(GT) and C6H6(GT) as one of the features to predict CO(GT). The same logic applies to NOx(GT) prediction. I include CO(GT) and NO2(GT) as one of the features and one of the features for predicting C6H6(GT) is CO(GT).

After identifying the potential parameters, I first format the date time into variable type date time. Then I extract year, month, day, day of week, day of year, and hour from it. I then create lagged features, which is the previous data of CO(GT), NOx(GT), and C6H6(GT). Lastly, I create rolling average and standard deviation for CO(GT), NOx(GT), and C6H6(GT) as the added features. After creating all these new features, I then pack these features and use them to train the model.

I start with a basic model, which is Linear regression first. Then, I try implementing a more advanced model like Random forest to compare the accuracy of these two models. I begin with a simple model and simple features. The first model I test is Linear regression with year, month, day, day of week, day of year, and hour as features. The mean squared error (MSE) and mean squared error (MAE) for this model are 1.90 and 1.16, 27612.65 and 116.51, and 48.70 and 5.22 for CO, NOx, and Benzene respectively. The MSE and MAE for CO, NOx, and Benzene predicted by Random forest without lagged features are 1.68 and 1.00, 4867.19 and 42.89, and 10.64 and 2.20. After fitting simple models, I try more advanced features and models. When fitting all the features to predict concentration level of CO, we can expect the accuracy of Linear regression at MSE = 0.00 and MAE = 0.00, and MSE = 0.07 and MAE = 0.15 for Random forest. The accuracy of models that predict concentration of NOx is MSE = 6909.13 and MAE = 60.48, and MSE = 4867.19 and MAE = 42.89 for Linear regression and Random forest respectively. For the concentration level of Benzene, Linear regression can predict the result with MSE = 15.11 and MAE = 2.52, while Random forest has MSE = 4.80 and MAE = 1.40. From these results, advanced features that include factors like lagged features and correlations improve the overall accuracy of the model. Moreover, Random forest seems to have a better accuracy in general. However, Linear regression performs better for CO prediction but it might overfit the dataset.

In addition, I tried fitting advanced models, like Seasonal Autoregressive Integrated Moving Average (SARIMA) and Long Short-Term Memory (LSTM). For SARIMA, I use the parameters about lag, decay, and seasonality from ACF and PACF. These 2 models fit the data pretty well since the dataset has the pattern of lagging relationships and seasonality. MSE and MAE for SARIMA are 2.93 and 1.43 for CO, 80984.81 and 244.84 for NOx, and 73.99 and 7.23 for Benzene, respectively. The MSE and MAE for CO, NOx, and Benzene are 0.54 and 0.50, 11105.12 and 70.77, and 9.60 and 2.06, respectively. The overall MSEs and MAEs are shown in Table 1. Different models work better for different pollutants. Linear regression with lagged features is the best model to predict CO(GT), while Random forest with lagged features provides the best prediction for NOx and Benzene. However, Linear regression with lagged features might overfit the data, so in general, Random forest with lagged features is the best model to predict air quality variations.

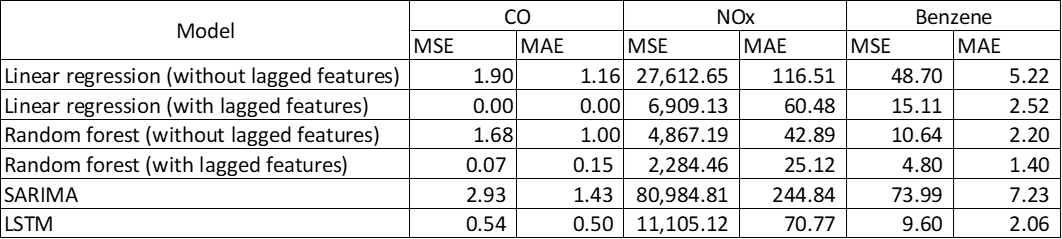


Table 1: MSE and MAE of different models

**Conclusion and limitations**

This project **s**uccessfully develops a real-time air quality prediction system using Apache Kafka and machine learning models. By simulating live data streams with Kafka, performing detailed preprocessing and exploratory analysis, engineering time-based and statistical features, and training predictive models, we were able to forecast concentrations of key pollutants such as CO, NOx, and Benzene with reasonable accuracy. Among the models tested, Random Forest delivered the best performance, especially for NOx and Benzene.

However, the project has several limitations. First, the dataset used is over a decade old (2004–2005) and based on data from a single city, which limits its generalizability to modern, global urban contexts. Second, while manual feature engineering produced effective results, it may miss latent patterns that deep learning models could capture more effectively. Third, the model was trained and evaluated on a relatively small temporal span (less than one year), restricting its ability to capture long-term seasonal effects. Forth, the models were trained only by a limited number of features due to the limited number of datasets. If I have more dataset that contains other attributes related to the prediction, I would be able to make a better prediction. Finally, the advanced model like SARIMA takes a lot of time and computation power to train. I tried varying the parameters of the model but it took a long time to process and some combinations of the parameters couldn’t run through, so I cannot improve the accuracy of that model. Also, the model takes some time to process, so I could not run a prediction one-by-one every time data is consumed. Kafka stops consuming messages. I have to implement the batch-streaming method. The messages are consumed in a batch and make predictions in each batch.