Forecasting Air Pollutant Levels Near Major Construction Zones in Downtown Toronto

by

Markus Kunej

Supervisor: Professor Greg J. Evans

April 2023

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

Air pollution is the greatest environmental burden to human health, contributing to over 8 million premature deaths every year. A major source of air pollution comes from the large construction zones resulting from urban densification. The increased emission from these zones causes health problems in construction workers and residents, such as increased cases of chronic obstructive pulmonary disease (COPD). It has also been shown that air pollutant emission rates vary with weather and construction activity. And so, with an accurate method of forecasting air pollutant levels in these construction zones, construction companies can schedule their activity in such a way to minimize their workers' and local residents' exposure to dangerous air pollutants. Past attempts at modelling emission rates in construction zones looked at each air pollutant individually and did not consider the effect of weather variables on the system, gaps which are addressed in this paper.

Air pollutant levels (CO, CO_2, NO, NO_2, O_3, PM_1, $PM_{2.5}$, and PM_{10}) and external data (humidity, pressure, temperature, wind direction, wind speed, and noise) were captured using 10 AirSENCE devices located around the Yonge & Eglington construction hub in downtown Toronto. All data was modelled together as a multivariate time series using both vector autoregressive (VAR) models and vector autoregressive-moving-average (VARMA) models. Various model parameters were tested, where it was observed that the VARMA model fitted with the largest training size (1000 data points) and lowest resolution (60-minute) data available outperformed all other models. It could predict air pollutant levels 1-day ahead with over 80% accuracy, but the 1-week and 2-week forecasts were less than 20% accurate. Sobol's sensitivity analysis was performed on the model, showing the existence of higher-order relationships between the various pollutants and weather variables. In addition, 5 out of the 8 air pollutants were most sensitive to changes in a weather variable, reinforcing their inclusion with modelling air pollutants. While a model with higher accuracy for its long-term forecasts would be required for construction activity scheduling, the work done in this paper is a good starting point for further progress or investigation into the complex relationships between air pollutants and weather in construction zones.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Greg Evans for the continuous support and ongoing mentorship. His feedback throughout the process was invaluable, and I’m extremely grateful he agreed to supervise me part-way through the year.

In addition, I would like to thank my thesis coordinator Alan Chong for his role in connecting me with Prof. Greg Evans.

I would also like to thank Nishitha Shashidhar for welcoming me and answering my questions regarding the data-hosting website.

Lastly, I would like to thank my family and friends for their love and support throughout my entire undergraduate studies.

Table of Contents

List of Figures

List of Tables

List of Equations

1 Introduction

Air pollution is the greatest environmental burden to human health. It contributes to over eight million premature deaths globally every year, of which 14,400 are in Canada [1]. Two ways people cause air pollution are constant use of an automobile for transportation and high energy consumption. Thankfully, both can be reduced through urban densification, the process of increasing the density of people living in urban areas. When implemented along with accessible public transit, urban densification reduces the distance people need to travel for work and leads to a lower average energy consumption due to the efficiency of high-density buildings [2]. It is also associated with reduced land take, which allows for greater availability of land for agriculture, nature, and biodiversity.

Urban densification often requires a major construction overhaul. This is especially true for projects that require deep pits to be dug, such as for high-rise buildings or underground transportation. These plans can lead to construction lasting up to a decade long [3]. As a result of the heavy machinery, increased congestion on the roads, and the release of particles (like dust) into the air, major construction sites lead to an increase in air pollutant levels. A case study in Qingyuan, China, found the daily concentration of total suspended particulates (TSP) near a construction site increased by 42.24%. It also saw a 16.27% increase in particulate matter 2.5 (PM2.5), which refers to particles that are 2.5 microns or less in width [4]. PM2.5 pose the greatest health risks, since due to their small size, they can lodge deeply into the lungs [5]. So, while there are long-term benefits to urban densification, the short-term decline in air quality caused by construction sites is worrying.

Air quality is popularly measured with the US Air Quality Index (AQI). This index considers the following pollutants: Ozone (O3), PM2.5, PM10, Carbon Monoxide (CO), Sulfur Dioxide (SO2) and Nitrogen Dioxide (NO2) [6]. Weather forecast providers display the AQI value for a given location, usually for an entire city. However, due to environmental and location-based factors, such as major construction, the given AQI value may not be representative of all sub-regions within a given location [4].

2 Background

2.1 Construction Sites Large Impact on Total Particulate Emissions

An article from the Harvard School of Public Health looked at different open sources of air pollution – those stationary sources too great in extent to be controlled through enclosure or ducting – and their overall contribution to national particulate emissions in 1976. Due to the lack of particulate measuring devices at the time, the authors instead constructed formulae to estimate the yearly Total Suspended Particulate (TSP) emissions from each of the open sources. They found that construction activities represented 3.8% (22 X 10^6 ton) of total particulate emissions from open sources [7].

2.2 Negative Health Effects

Over 300,000 Swedish male construction workers were followed and analyzed from 1971 to 1999. They found that the fraction of chronic obstructive pulmonary disease (COPD) among the exposed attributable to any airborne exposure was estimated as 10.7% overall and 52.6% among people who have never smoked. This means that of all the workers who developed COPD and never smoked before, 52.6% of the cases were due to exposure to airborne emissions [8]. While this study only looked at construction workers' health, residents in dense urban areas undergoing construction would also be exposed to similar airborne particles.

2.3 Variability from Construction Activity and Weather

An article from 1999 looked at estimating the overall emission rate of total suspended particulates (TSP) caused by road construction in Taiwan. They found that dust emissions often varied substantially from day to day depending on the level of activity, the specific operations and the prevailing meteorological conditions, making it difficult to assess the total contribution of such emissions to the air pollution levels of a city or region [9]**.**

Since the current measuring devices near Yonge and Eglinton capture real-time meteorological and noise values (which can be used to estimate construction activity), there exists an opportunity to address the variability this study brings up in understanding emission rates.

Also, in a 2014 study of road work construction in London, England, PM_{10} increased during the construction period up to 15 $\mu g\ m^{-3}$ during working hours compared to concentrations before the road works. This breached the EU PM_{10} limit value (LV) at the time [10]. This demonstrates the magnitude that construction activity has on resulting air pollutant levels, so-much-so that it can breach local environmental guidelines.

2.4 Ontario Environmental Protection Act and Toronto Dust Control by-law

In Ontario, businesses must follow the air quality regulations set out by the Environmental Protection Act [11]. These include limits to particulate matter emissions. Penalties can range from $1,000 per day for less serious violations such as failure to submit a quarterly report, to $100,000 per day for the most serious violations, including a spill with a significant impact.

In Toronto, City Council has decided that the Dust Control by-law will not apply to municipal works, construction occurring on commercial and industrial properties, nor will it apply to the construction of a multi-residential building, subdivision, or mixed-use development [12].

This is relevant to this thesis, as we are looking at the air pollutant levels in large construction hubs in downtown Toronto. These construction sites are likely some of the biggest dust producers yet are given an exemption from the city.

2.5 Previous Attempt to Model Air Pollutant Levels near Construction Sites

A 2022 study aimed to find the correlative relationship between seven air quality indicators (i.e., the air quality index, PM2.5, PM10, O3, NO2, SO2, and CO) and the number of construction sites in Hangzhou, China. They used non-linear regression models to estimate the number of construction sites. Then, based on guideline values of air parameters provided by the Chinese criteria and standard, the recommended maximum number of construction projects can be defined. The model did not generalize to other districts, however, and the lack of meteorological data was considered a limitation that needs to be addressed [13].

2.6 Methods for Multivariate Time Series Analysis

Unlike univariate time series, which look at a single variable over time, multivariate time series feature a vector of *k* variables observed over time. The goal with modelling multivariate time series is to accurately describe the relationship amongst the *k* variables and how they impact each other's future values. The vector autoregressive and moving average models are both successful modelling techniques to use for multivariate time series problems. Vector autoregressive (VAR) and vector autoregressive-moving-average (VARMA) models all require the time series variables to be stationary (i.e. not show seasonality) [14].

Vectorized Auto-Regressive (VAR) Model

In the VAR model, each variable has an equation modelling its evolution over time. It includes the variable's lagged (past) values, the lagged values of the other variables in the model, and an error term. In general, a *p*th-order VAR refers to a VAR model which includes lags for the last *p* time periods, and can be written as:

Equation 1: VAR Equation for a Single Variable

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + e_t ,$$

Where *c* is a *k*-vector of constants (intercept of the model), A_i is a time-invariant (*k x k*)-matrix, y_{t-i} are that variable's value *i* time periods earlier, and e_t is a k-vector of error terms [14].

Vectorized Auto-Regressive Moving-Average (VARMA) Model

Equation 2: VARMA Equation. AR Component Highlighted in Red, MA Component in Blue

$$y_t = \nu + \boxed{A_1 y_{t-1} + \cdots + A_p y_{t-p}} + Bx_t + \epsilon_t + \boxed{M_1 \epsilon_{t-1} + \ldots M_q \epsilon_{t-q}}$$

This model expands upon the VAR model through the addition of a moving average component to the equation. A moving average model (VMA) uses past error (or residual) terms to predict future values, unlike an autoregressive model which uses lagged values. A VARMA model combines both components in its equation, making it a more general version of a VAR and a VMA model. It historically performs better than one of those models separately because it combines the VAR model's ability to build relationships between the various variables and the VMA model's ability to capture long-term trends within each of the variables.

Sobol Sensitivity Analysis

Sensitivity analysis studies how the uncertainty in the output of a mathematical model can be divided and attributed to different sources of uncertainty in its inputs. Due to the large multivariate system attempted in this paper, sensitivity analysis is a useful tool to better understand how different air pollutants and weather variables affect each other, especially in combination with other variables. An example of this analysis is called Sobol' method, a variance-based sensitivity analysis technique. It decomposes the variance of the output of the model or system into fractions that are connected to inputs or sets of inputs. For instance, given a model with 2 inputs and 1 output, it might be found that 60% of the output variance is caused by variance in the first input, 20% by the variance of the second, and 20% because of interactions between the two (second-order variances). These percentages are interpreted as measures of sensitivities [15].

These sensitivities can be calculated through the generation of carefully selected sequences of inputs, and then observing the resulting outputs of the system. Sobol' method utilizes the Sobol' sequence, a popular quasi-random low-discrepancy sequence to generate uniform samples of a parameter space. For this paper, we will be using the Saltelli sampling scheme instead [16]. It extends the Sobol' sequence in a way to reduce the error rates in the resulting sensitivity calculations.

2.7 Research Questions

Through review of past literature and identifying the gaps in research, the following two research questions that will guide the thesis have been determined:

1. What effect do the active working hours and meteorological conditions have on the daily air pollutant levels near major, urban construction zones?

2. Using weather forecasts and past air pollutant values, can future air pollutant levels near major, urban construction zones be accurately predicted?

3 Methods

3.1 Obtaining the Data

The data for this project was collected by 10 AirSENCE devices deployed around the Yonge and Eglington construction hub in downtown Toronto.

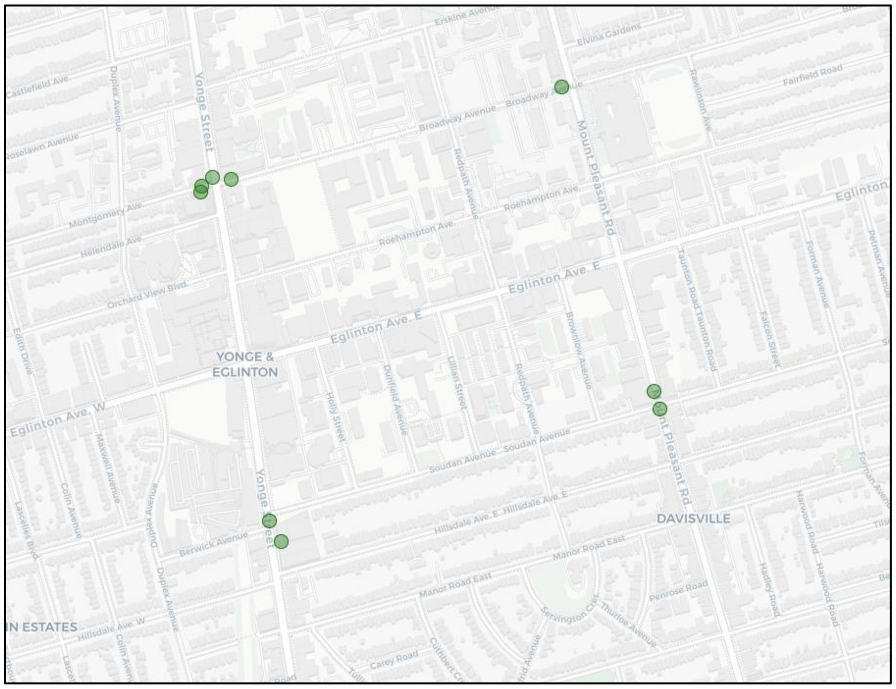


Figure i: Locations of the 10 AirSENCE Devices in Downtown Toronto

These devices were developed by A.U.G. Signals, a Toronto-based company specializing in multi-sensor systems. They recorded measurements in micrograms per cubic metre (μg/m^3) for the following pollutants:

Table 1: The 8 Pollutants Measured by the AirSENCE Devices

Pollutant Name	Referred to as
Carbon Dioxide	CO_2
Carbon Monoxide	CO
Nitrogen Dioxide	NO_2
Nitrogen Monoxide	NO

Ozone	O_3
Particulate Matter < 1 µm in size	PM_1
Particulate Matter < 2.5 µm in size	$PM_{2.5}$
Particulate Matter < 10 µm in size	PM_{10}

As well as measurements on other weather and external variables:

Table 2: Additional Variables Measured by the AirSENCE Devices

Variable Name	**Units Measured**
Air Quality Health Index (AQHI)	Scale from 1 (low risk) to 10+ (high risk)
Elevation	metres
Humidity	Relative Humidity (%)
Latitude	degrees
Longitude	degrees
Noise	dB
Pressure	hPa
Temperature	C°
Wind Direction	degrees
Wind Gust	m/s
Wind Speed	m/s

These 10 devices were installed on July 22nd 2022, and have been recording data since. Measurements for all variables were taken with a resolution of one minute, but the website used to receive and download the data (*airsencetech.grafana.net*) allowed for additional resolution options up to 60 minutes. Therefore, a range of time resolutions (1m, 15m, and 60m) were downloaded to determine the best resolution to use for long-term forecasting. Data could only be downloaded from the website as individual CSV files, each representing a single measured variable and monthly period (I.e. CO_2 for September 2022). Further processing and combining of the data would be required to model the data as a multivariate time series problem.

3.2 Data Processing and Cleaning

The downloaded data required some cleaning and processing before it could be used in model training and forecasting. The Python Data Analysis Library, *Pandas*, was the main tool used for this process. Firstly, the average measurement of a variable at a given time across the 10 devices was calculated (see Appendix A.1). This was done to reduce the impact of outliers or NaN values in the data, which would periodically occur for a device whenever it was rebooted or experienced a malfunction. Next, the individual variables had to be combined as a single, large Pandas DataFrame. This required re-indexing using date-time to ensure that each variable's data is correctly aligned with the others during concatenation.

Once the variables were combined into a single DataFrame, the data for a 40-day period was visualized (see Appendix B.1)., The *Latitude*, *Longitude*, and *Elevation* data were constant, since the devices were placed within the same block as one another and at a similar height. Because of this, they provide no additional information that would be helpful in modelling the pollutants, and were removed from the data to reduce the number of dimensions. The AQHI column was also removed, since this value is calculated using O_3, NO_2, and $PM_{2.5}$ (see Equation 3), all of which are already included as variables. Any effect AQHI has on future pollutant levels will also be seen in the individual variables that make up the value, so removing it can simplify the multivariate time series further.

Equation 3: Air Quality Health Index Calculation [17]

$$AQHI = (\frac{1000}{10.4}) \times [(e^{0.000537 \times O_3} - 1) + (e^{0.000871 \times NO_2} - 1) + (e^{0.000487 \times PM_{2.5}} - 1)]$$

Looking at a longer period of data from July 2022 – January 2023, we can see there was a reboot of the devices on October 3rd that caused missing CO_2 readings from all 10 devices for a period of 24 hours. Other variables were also affected, with several of them showing an abnormal spike in value (see Appendix B.2). Because of the missing values from all 10 devices, CO_2 would've had no value, making the data ineligible to be used for model training. The outliers in the other variables would have also impacted the training results. If it was desired to use the entire data period in model training, then statistical techniques would've been needed to remove outliers and estimate the missing CO_2 values. But for this project, it was simpler to remove the affected data, splitting the dataset into two sections (July 22nd – Oct 3rd and Oct 4th – Jan 1st, see Appendix A.3).

Another phenomenon in the data that required cleaning was when a device sensor malfunctioned in such a way, that instead of NaN values being recorded, it would incorrectly measure '0'. To avoid these 0's from affecting the average across the devices, they were replaced with NaN values (see Appendix A.4). This was only done for variables where a '0' measurement would not make sense, like CO_2 or pressure. Variables like temperature were untouched since 0°C readings are reasonable and expected in the fall and winter months. There were also scenarios where a device's sensor repeatedly recorded the same value for a variable. While it is possible for a measurement to remain unchanged for a couple of readings, there were cases where a device's readings for a variable kept constant over several hours, a statistically improbable event that suggests a device sensor issue. To detect and eliminate this, any value in the data that repeats itself for more than 10 consecutive measurements was converted to NaN values (see Appendix A.4).

By using the above data processing and cleaning techniques, biases and outliers in the data were reduced, improving the chances that a more accurate model can be produced.

3.3 Making the Data Stationary

One of the assumptions for the later tests and statistical models is that the data is *stationary*. A time series is considered stationary if there exists no trend in the data, a constant variance and constant autocorrelation structure over time, and no periodic fluctuations (seasonality) [18]. This assumption is necessary, since most time series models assume each point is independent of one another, and that the statistical properties of a system do not change over time [19]. A common statistical test to determine whether a time series is stationary is the Augmented Dickey Fuller (ADF) test [20]. This is an expanded version of the Dickey-Fuller test, a unit root test that tests the null hypothesis that α = 1 in the following equation:

Equation 4: Dickey-Fuller Model Equation [20]

$$y_t = c + \beta t + \alpha y_{t-1} + \phi \Delta Y_{t-1} + e_t$$

Where, α is the coefficient of the first lag on Y, y_{t-1} is the first lag of the time series, ΔY_{t-1} is the first difference of the series at time t-1, c and β are constants, and e_t is an error term. If α = 1, it implies the presence of a unit root, and therefore the time series is non-stationary.

The ADF test includes a high order regressive process in the model (see Equation 5).

Equation 5: Augmented Dickey Fuller Model Equation [20]

$$y_t = c + \beta t + \alpha y_{t-1} + \phi_1 \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} + \cdots + \phi_p \Delta Y_{t-p} + e_t$$

The only change is the addition of more differencing terms, which adds more thoroughness to the test. The null hypothesis is the same as the Dickey Fuller test. It is important to note that the p-value obtained should be less than the significance level (0.05) in order to reject the null hypothesis. That is, the series is stationary if the p-value < 0.05.

The ADF test was done on the collected data using statsmodels' adfuller Python function [21][22]. This version includes an "autolag" parameter, which chooses the number of lags (p in Equation 5) that minimizes the Akaike information criterion (AIC), an estimator of model quality [23]. The test was done on all three of the time resolution datasets for both the July 22nd -Oct 3th and the Oct 4th – Jan 1st series (in total, 6 multivariate time series). The majority of variables were determined to be stationary, with only 2 instances of a variable failing to reject the null hypothesis (i.e. they are non-stationary. See Table 3).

Table 3: ADF Test Results of Non-Stationary Variables

Variable	**Dataset**	**ADF Test Statistic**	**P-Value**	**# of Lags Used**	**# of Observations Used**	**Conclusion (P-Value < 0.05?)**
Temperature	60m resolution, July-Oct	-2.84	0.052	24	1709	Non-Stationary
Pressure	1m resolution, July-Oct	-1.94	0.31	69	103943	Non-Stationary

Non-stationary series must undergo a transformation to be converted to a stationary one. The most common transform to do this is the *difference transform* [18]. This consists of taking the data point at the current time and subtracting it with the point before [19]. Given the series Z_t, we create the new series:

Equation 6: Difference Time Series Data [18]

$$Y_i = Z_i - Z_{i-1}$$

The result is a series of differences between points at time i. By removing the changes in the level of a time series, the trend and seasonality of the data are eliminated, making it stationary.

The first-order difference of the 2 non-stationary variables in Table 3, were calculated, and the ADF test was re-run (see Appendix A.5). Both series were now stationary, and further differencing was not required (see Table 4). All datasets were now ready to for statistical model analysis.

Table 4: ADF Test After Difference Transform

Variable	**Dataset**	**ADF Test Statistic**	**P-Value**	**# of Lags Used**	**# of Observations Used**	**Conclusion (P-Value < 0.05?)**
Temperature	60m resolution, July-Oct	-8.84	1.7E-14	23	1709	Stationary
Pressure	1m resolution, July-Oct	-25.7	0.00	69	103943	Stationary

3.4 Train Vector Autoregressive Models

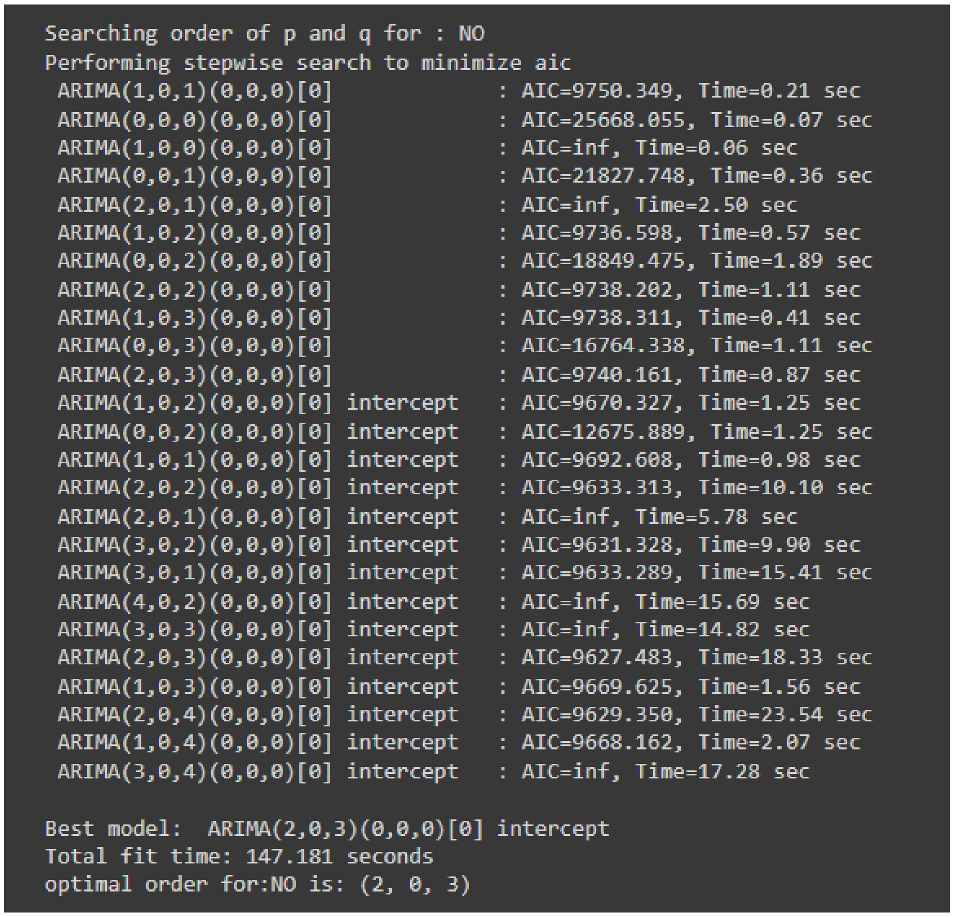
To train a vector autoregressive model, the first step was to determine the best lag order for the model. This refers to the number of past values (p) that should be considered for each of the variables in the model equations. The statsmodels Python library was used for this task, specifically the *statsmodels.VAR.select_order* function [24]. This function determines the AIC value for a multivariate time series for each of the # of lags from 0 to *maxlags* (input parameter to the function), selecting the lag order that produced the lowest AIC value (estimation of prediction error). For this project, *maxlags* was set to 10, since VAR models with a lag order higher than 10 were too computationally expensive to train for this project's environment (Google Colab) [Appendix A.6].

After determining the lag order, the VAR model was fitted to the multivariate time series. The *statsmodels.VARMAX.fit* was used, setting the p value to the determined optimal lag order, and the q order (Moving Average parameter) to 0, since there is no MA component in a VAR model [25][Appendix A.6]. The output is a fitted VAR model, which includes the coefficients for the autoregressive equations.

3.5 Train VARMA Models

Unlike a VAR model, which has only one parameter (lag order p) to determine, vector autoregressive-moving-average models need to determine the best lag order and moving average component, q, to use. This was achieved through a stepwise search determining the best p and q combination for each individual predicting pollutant, fitting a VARMA model for each of these unique p and q combinations, and then selecting the pair that produces a model with the lowest prediction error.

Figure ii: Example of the Stepwise Search of p and q for NO



```
Searching order of p and q for : NO
Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=9750.349, Time=0.21 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=25668.055, Time=0.07 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=inf, Time=0.06 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=21827.748, Time=0.36 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=inf, Time=2.50 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=9736.598, Time=0.57 sec
 ARIMA(0,0,2)(0,0,0)[0]             : AIC=18849.475, Time=1.89 sec
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=9738.202, Time=1.11 sec
 ARIMA(1,0,3)(0,0,0)[0]             : AIC=9738.311, Time=0.41 sec
 ARIMA(0,0,3)(0,0,0)[0]             : AIC=16764.338, Time=1.11 sec
 ARIMA(2,0,3)(0,0,0)[0]             : AIC=9740.161, Time=0.87 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=9670.327, Time=1.25 sec
 ARIMA(0,0,2)(0,0,0)[0] intercept   : AIC=12675.889, Time=1.25 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=9692.608, Time=0.98 sec
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=9633.313, Time=10.10 sec
 ARIMA(2,0,1)(0,0,0)[0] intercept   : AIC=inf, Time=5.78 sec
 ARIMA(3,0,2)(0,0,0)[0] intercept   : AIC=9631.328, Time=9.90 sec
 ARIMA(3,0,1)(0,0,0)[0] intercept   : AIC=9633.289, Time=15.41 sec
 ARIMA(4,0,2)(0,0,0)[0] intercept   : AIC=inf, Time=15.69 sec
 ARIMA(3,0,3)(0,0,0)[0] intercept   : AIC=inf, Time=14.82 sec
 ARIMA(2,0,3)(0,0,0)[0] intercept   : AIC=9627.483, Time=18.33 sec
 ARIMA(1,0,3)(0,0,0)[0] intercept   : AIC=9669.625, Time=1.56 sec
 ARIMA(2,0,4)(0,0,0)[0] intercept   : AIC=9629.350, Time=23.54 sec
 ARIMA(1,0,4)(0,0,0)[0] intercept   : AIC=9668.162, Time=2.07 sec
 ARIMA(3,0,4)(0,0,0)[0] intercept   : AIC=inf, Time=17.28 sec

Best model:  ARIMA(2,0,3)(0,0,0)[0] intercept
Total fit time: 147.181 seconds
optimal order for:NO is: (2, 0, 3)
```

The stepwise search utilized the *auto_arima* method from the *pmdarima* library [26]. It accepts a *max_p* and a *max_q* parameter, which was chosen to be 5 for both [Appendix A.7]. This was because allowing for larger *p* or *q* values would create a model that was too computationally expensive to fit.

Another difference between finding *p* with a VAR model, is the *statsmodels.VAR.select_order* method for finding just *p* considers a multivariate time series when calculating the AIC scores. However with *auto_arima*, the stepwise search can only be performed on univariate time series (only one variable). So, after determining the best *p* and *q* value for the individual predicting pollutants, their affect on the other variables of the system and the overall forecasting accuracy of the multivariate model need to be considered. This was done by fitting (training) a VARMA model using each of the unique combinations of *p* and *q*, and then calculating the average mean absolute percentage error (MAPE) across all 8 of the predicting pollutants [Appendix A.7]. The fitted model which produced the lowest average MAPE score was deemed the best overall model for multivariate time series, and its *p* and *q* values were selected.

3.6 Selecting Different Parameters to Train

Based on the initial goal of finding the best performing model of the multivariate time series air pollutant data, there were several model parameters that needed to be explored. This included the model type (VAR vs VARMA), training data size, and data resolution. Due to the extended period of time it took to fit certain models, it was decided to explore these parameters with an iterative process. For example, after the highest performing model type was determined, the future models being trained to determine the best dataset size would all use the highest performing model type. This greatly reduces the number of combinations of model parameters to try, and allows us to reach our desired model parameters quicker. With this goal in mind, the following models were selected for training:

Model Type

To test the effect that the model type had on the accuracy of the model, two models were trained using the same dataset (Oct-Jan), same resolution (60m), and same training size (1000 data points). However, one was fitted to a VAR model, while the other was fitted to a VARMA model.

Training Size

Once the best model type was determined (VARMA), this was used along with a constant data resolution (15m) and period (Oct-Jan) to train 4 models of various training dataset sizes: 100, 300, 500, and 1000 points. A “point” of data is a vector of the recorded measurements for each of the variables at a given time, and is dependent on the resolution of data. In this case, using a 15m resolution, 1000 points represents 250 hours of data. A longer training size was attempted, but was too computationally expensive to complete training.

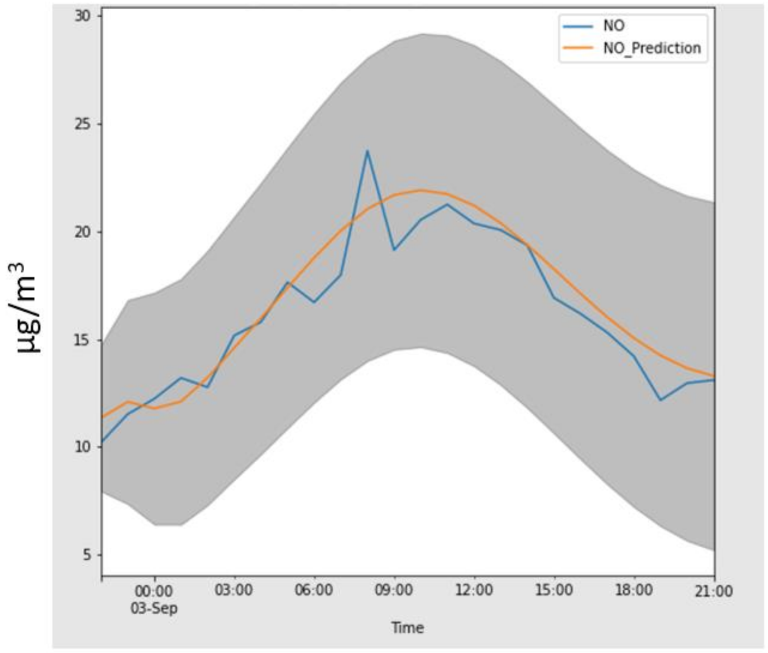
<u>Data Resolution</u>

Finally, with both the best model type (VARMA) and training size (1000) determined, a model was trained for each of the 3 data resolutions downloaded. This included a 1m, 15m, and 60m resolution.

The trained models were all saved and recorded throughout the project. For a full list, including the parameters used to fit them, see Appendix C.

3.7 Plotting Forecasts

Figure iii: Example Plot of 1-Day Forecast of NO



To better visualize the forecasting ability of the models, their predictions were plotted against the real pollutant levels. To do this, predictions one-day, one-week, and 2-weeks ahead were generated on never-before-seen data for the model. The 95% confidence interval of these predictions were also calculated (see Appendix A.8). Then, using the *matplotlib* library, display each forecast range against the recorded pollutant level in the test data. Include the 95% confidence interval as a shaded section and save the figures (see Appendix A.9). An example of this plot for a 1-day forecast of NO can be seen in Figure iii.

3.8 Metrics used to Compare Models

To compare the forecasting ability of different models, one must choose a set of metrics to judge the model's accuracy. One of the most common metrics used for comparing time series forecasts is the root mean square error (RMSE). However, since we are considering different variables, we must use a percentage-based error metric. Otherwise, variables with a much larger mean value (like CO_2) would naturally produce significantly higher error values, creating a bias when taking the average error across all variables. So, it was decided to use the mean absolute error percentage (MAPE), one of the most common percentage-based errors, instead. The calculation is as follows,

Equation 7: Mean Absolute Percentage Error Calculation [27]

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|A_i - F_i|}{A_i}$$

Where A_i is the actual value, F_i is the forecast value, and n is the total number of observations. MAPE was then used to compare a group of models by plotting the errors for each individual pollutant, and then the average MAPE across all pollutants for each of the models (see Appendix A.10).

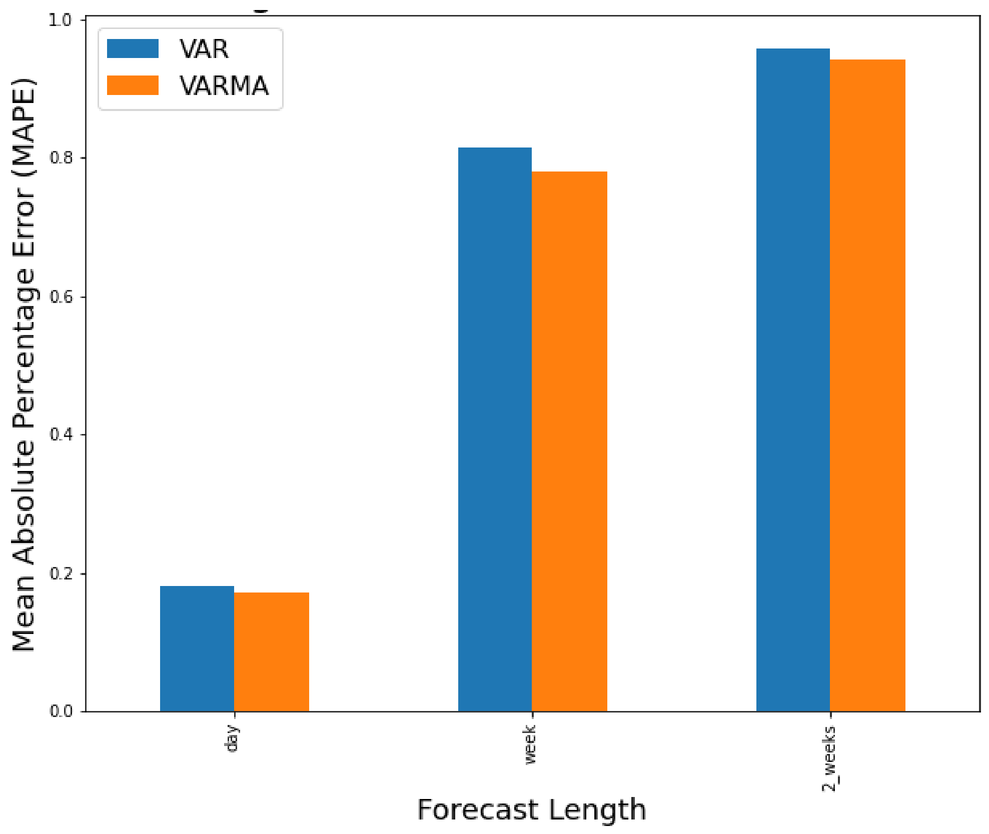
3.9 Sobol Sensitivity Analysis Implementation

The Sobol Sensitivity Analysis was performed on the most accurate model found (VARMA, 1000 data points, 60m resolution), with the help of *SALib*, a sensitivity analysis library in python [28]. First, model inputs were generated using Saltelli's extension of the Sobol sequence. Next, using the generated sequence as inputs for the model, predictions were made for each of vector inputs. Finally, using the *analyze_sobol* function from SALib, the first, second, and total-order sensitivity indices for each of the air pollutants were calculated (see Appendix A.11). These indices were then visualized using heatmaps.

4 Results and Discussion

4.1 Addition of Moving Average Component Improves Model

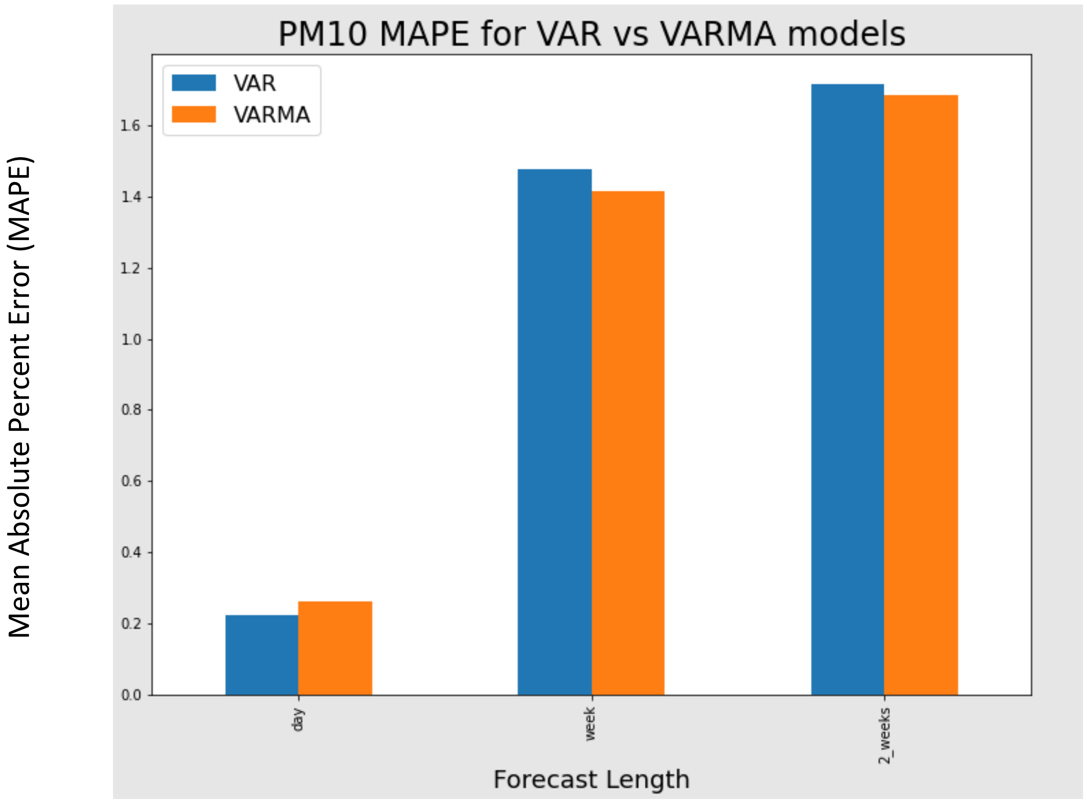
Figure iv: VARMA Model Outperforms VAR Model on Average



Using a training size of 1000 data points from the Oct-Jan dataset and a resolution of 60m per datapoint, the vector autoregressive model with a moving average component (VARMA) slightly outperformed the vector autoregressive model (VAR) when looking at the average MAPE across all predicted pollutants (see Figure iv). A possible reason for this is that the addition of the moving average component allows for the generation of future values based on the errors of past forecasts, instead of purely relying on past values like an autoregressive model. In a VAR model, there is an ignorance of correlated noise structures (which are unobservable) in the time series. These imperfectly predictable terms in current time and previous steps (past predictions) can also be informative for predicting future observations, something the moving average component takes into consideration [29].

Another benefit the VARMA model had for this project is its ability to approximate a time series correlation structure with a much lower AR order (p) and MA order (q) than the p order required for a purely VAR model. This is because in practice, it is often the case that high-order VAR models can be well approximated by low-order VARMA models [30]. Originally for this project, the maximum p order allowed when selecting the best order while training a VAR model was set to 20. But due to the computational complexity of determining all coefficients in a high-order VAR model with 14 variables, the model was unable to fit after more than 24 hours of training, and so the maximum order had to be reduced to 10. Because of these computational limits reducing the maximum orders, and the VARMA model's "efficiency" at producing an accurate model with low-order values, the VARMA model was more suited for this specific multivariate time series problem.

Figure v: VAR Outperforms VARMA on Daily Forecasts for Particulate Matter



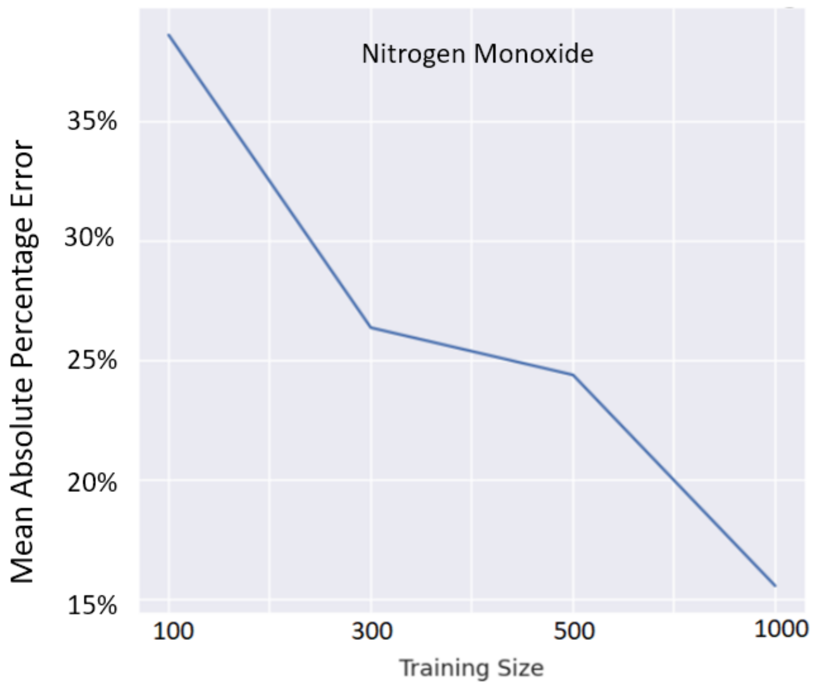
It is interesting to note that the MAPE of the particulate matter variables (PM_1, $PM_{2.5}$, PM_{10}) was more alike between the VAR and VARMA model, and in some cases the VAR outperformed VARMA for the daily forecast (see Figure v). This suggests that the particulate matter data may have a strong autoregressive component and/or no significant moving average component. Another reason could be that the VARMA model overfit the particulate matter data, since VARMA models have more paramters to estimate than VAR models, and the sample size was small for the problem (at least a couple of years worth would be preferred).

The full MAPE graphs for each individual pollutant comparing a VAR vs VARMA model can be found in Appendix D.

4.2 Larger Training Dataset Size Decreases Model Error

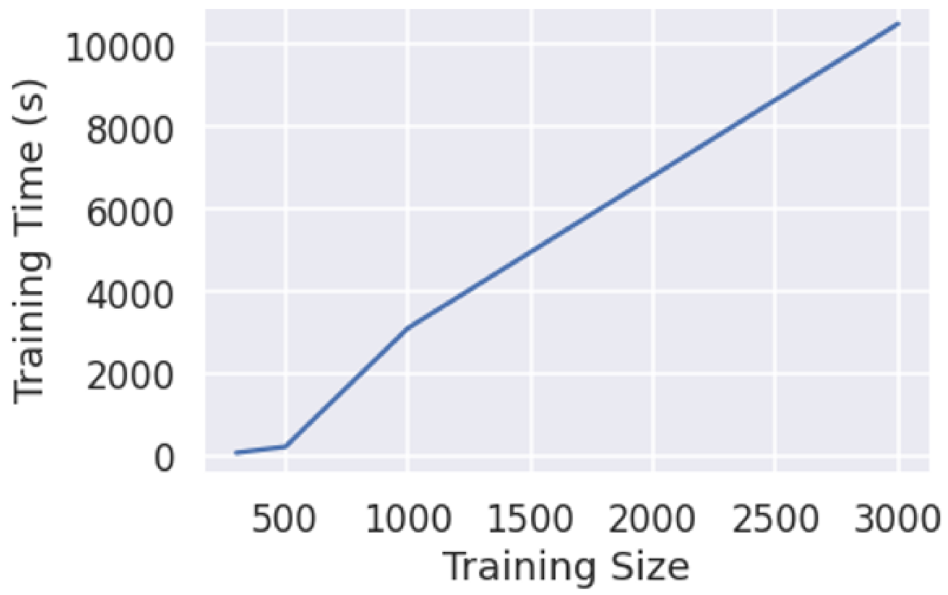
Using VARMA models trained on the Oct-Jan dataset with a resolution of 15m, it was observed that as the training size (number of data points used in model fitting) increased, the MAPE for a week-long forecast generally decreased (see Figure vi). This is likely because air pollutants follow diurnal patterns, and the longer the training period, the more examples of these daily changes the model can learn from while fitting. For example, 100 data points at a 15-minute resolution is only a 25-hour period, barely over a single day, while 1000 data points equates to a 250-hour period.

Figure vi: Increasing the Training Size Reduces Model Error



However, the largest training period used (250 hours) was still well below the training size desired. To cover long-range / seasonal patterns commonly exhibited by air pollutants, a training period of a year or more would have been preferred [31]. But increasing the training size requires more computations during training, and thus took longer to fit the model. This can be seen in Figure vii, where there was a linear increase in the time it took to fit a VAR using varying training sizes. Due to the large number of parameters required to learn during training for a 14 variable VARMA model, 1000 data points was the maximum size that could be used before the model was unable to complete training before timing out on Google Colab.

Figure vii: Increasing the Training Size Increases the Training Time

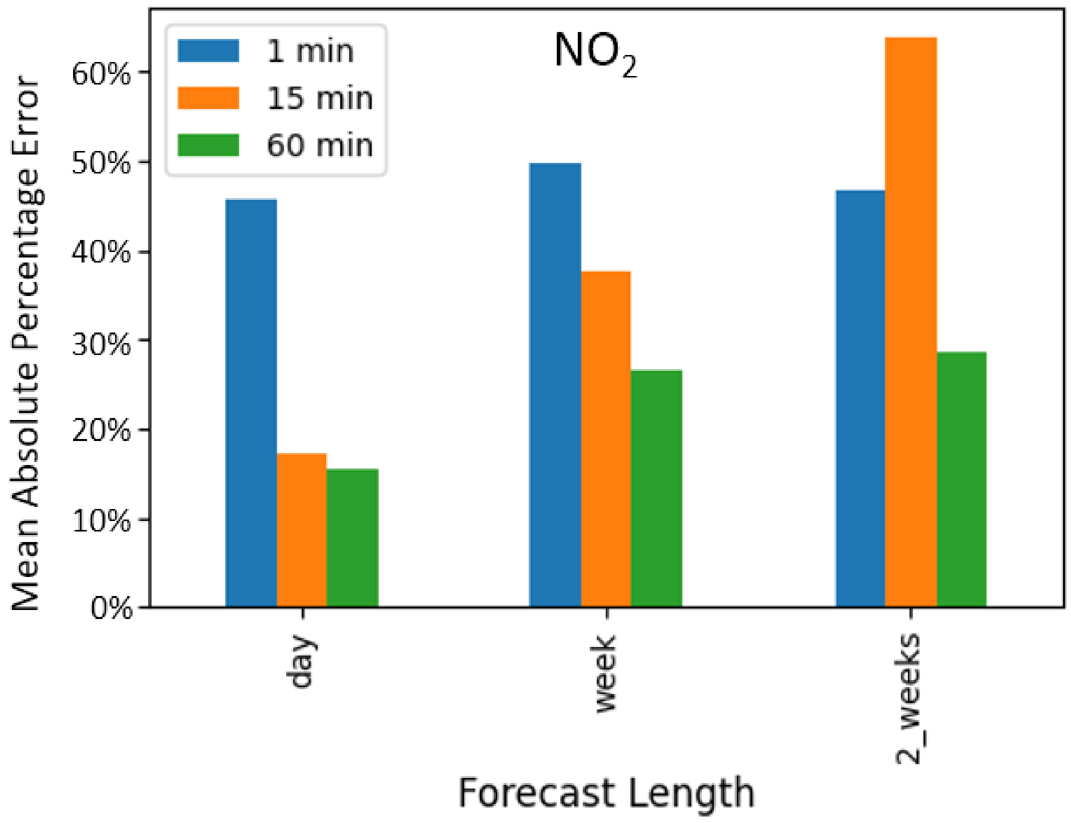


Plots showing MAPE for various training sizes for every pollutant can be found in Appendix E.

4.3 Lower Resolution Data Improves Model's Predictions

Using the previously determined model type (VARMA) and training size (1000 data points), data with different resolutions were used to train several models. It was observed that the model fitted using the lowest resolution (60 min) data predicted future air pollutant levels with less error than the models fitted using higher resolution (15 min, 1 min) data (see Figure viii).

Figure viii: Lower Resolution Data Produced Less Error for All Forecast Lengths



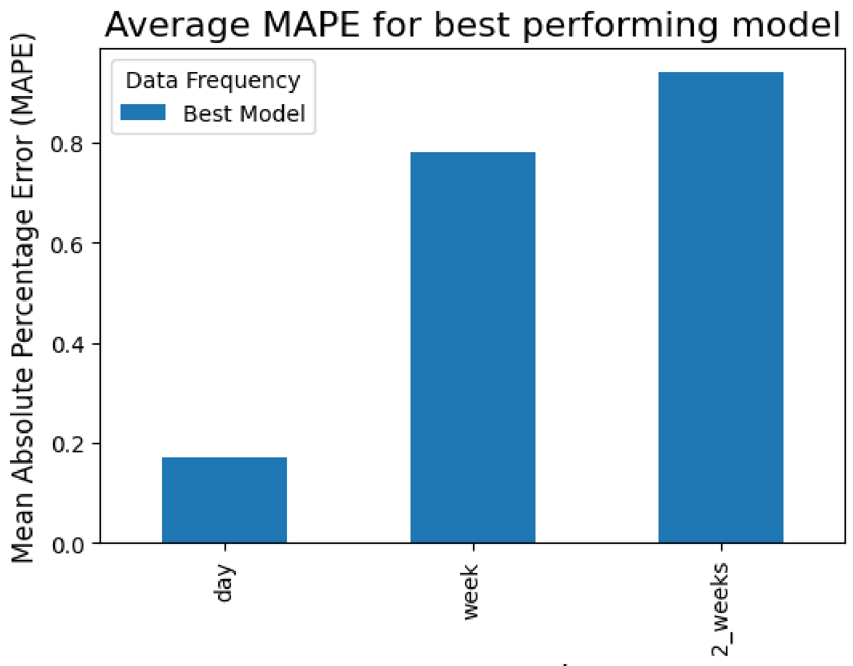
Similar to how increasing the training size allowed for a larger training window, using lower resolution data also allows for a longer time period to be used for fitting. 1000 data points at a 1-minute resolution is under 17 hours of data, but at a 60-minute resolution it is over 40 days long. Because the 1-minute resolution data does not include a full day's worth of data, it cannot teach a model of the common diurnal patterns seen in air pollutants. This explains the poor predictive ability for a 1-day forecast (~45% MAPE) of the 1-minute resolution model compared to the 15 and 60-minute resolution models, which both use data covering at least a 10-day period (see Figure viii).

It is interesting to note that the 15-minute and 60-minute resolution models performed similarly for the 1-day forecast, but as the forecast length increased, the 60-minute model performed significantly better (see Figure viii). This further enforces the idea that the longer time period with lower resolution data is more important in model training then the increased precision in higher resolution data. The 15-minute resolution data is just over 10 days long, so there would be *less* examples of possible weekly patterns in air pollutant levels than in the 60-minute resolution data, which covers more than 40 days. 10 days is also not enough to show the existence of potential bi-weekly changes either. For a comparison of MAPE with different data resolutions for all the air pollutants, see Appendix F.

4.4 Best Performing Model

After determining the model type (VARMA), training size (1000), and data resolution (60-minutes) that reduced the mean absolute percentage error in predicting air pollutant levels, these parameters were all used in fitting an "optimal" model. The forecasting performance of the model was tested and the average MAPE across all pollutants are shown in Figure ix.

Figure ix: Average MAPE of Best Model



The 1-day forecasts were highly accurate (80%+), producing a MAPE of less than 20%. This is an encouraging result, suggesting the model was able to learn of the diurnal patterns commonly seen in air pollutants (see Figure x). However, the longer-term forecasts were significantly less accurate, with the weekly and bi-weekly forecast averaging a MAPE of over 80%. To understand why, the weekly forecast of a pollutant (NO) was plotted and observed (see Figure xi). It can be seen that the prediction (in orange) “flattens”, in the sense that it trends towards the historic mean of the pollutant. This phenomenon was noticed in the long-term forecasts of all predicted pollutants (see Appendix H.1). While the exact reasoning for this occurrence is unknown, one possible explanation is that the training data needed to cover a longer time period for the model to better understand long-range patterns. When a model is in doubt, often trending towards a variable’s mean will produce less error than attempting to follow a pattern more precisely.

Figure x: Daily Forecast of NO Follows Diurnal Pattern

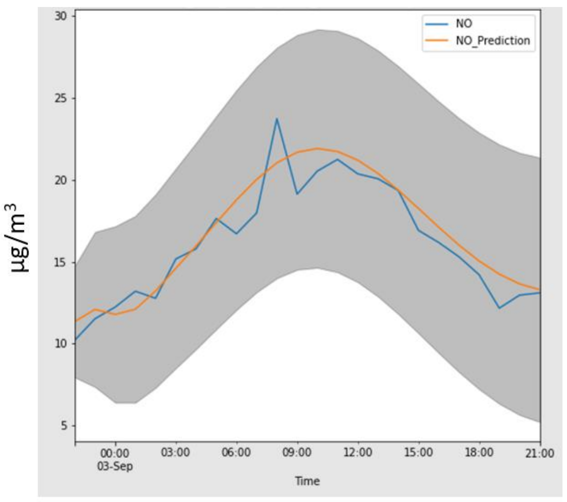
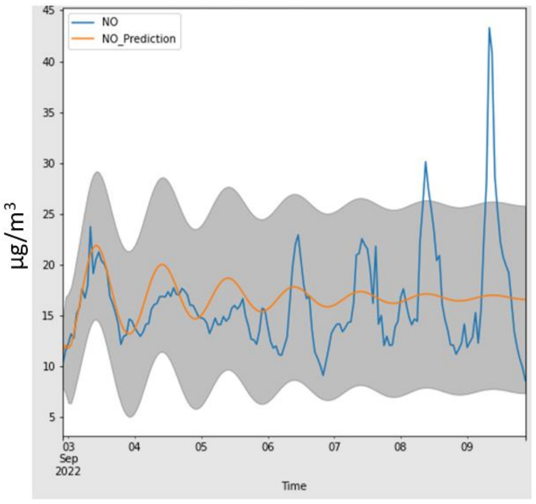


Figure xi: Weekly Forecast Trends Towards Historic Mean



Another possible explanation could simply be the difficulty of the task-at-hand, due to the high levels of “randomness” that exists with predicting weather and air pollutant levels. Since this model combines both the weather and air pollutant variables into a single large multivariate time series problem, it attempts to model the weather variables in addition to the pollutants. While this approach gives the model the most freedom in discovering hidden relationships between the variables, it also greatly increases the complexity of the system it’s trying to understand. For example, even if the model accurately discovers how a weather variable can impact a resulting air pollutant’s level, it must also be able to correctly predict future levels of that same weather variable in order to make an accurate air pollutant prediction. Small inaccuracies in the model will amplify over time, as it makes future predictions based on its own past predictions.

4.5 Sensitivity Analysis Findings

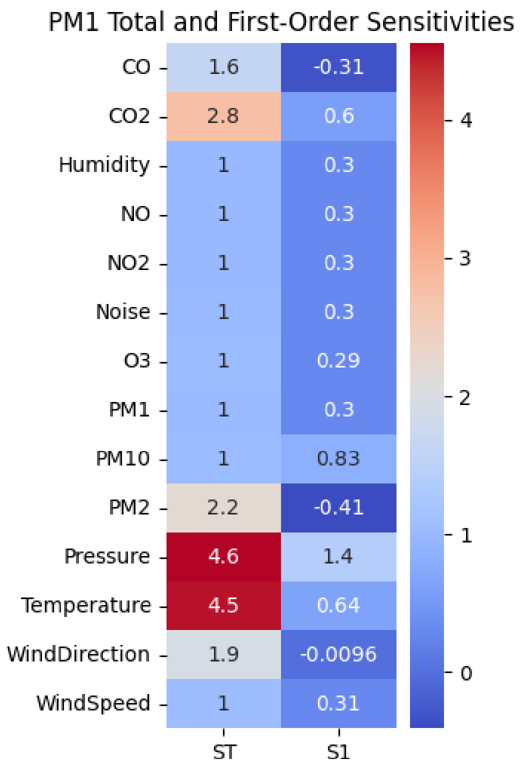
With the most accurate model determined to be of type VARMA, with a training size of 1000 data points and a resolution of 60 minutes, Sobol sensitivity analysis was performed on it. The total, first-order, and second-order sensitivities for each air pollutant were obtained, and can be found in Appendix F. Positive sensitivity indices indicate that as the input parameter increases, the output variable will either increase or decrease consistently. Likewise, negative sensitivity values indicate that changes in the input parameter have no effect on the output variable.

Table 5: Top Total Sensitivity for each Air Pollutant

Responding Pollutant	Top Total Sensitivity
PM1	Pressure
PM2	CO2
PM10	Pressure
CO	Wind Direction
CO2	Humidity
O3	CO
NO	NO2
NO2	Wind Direction

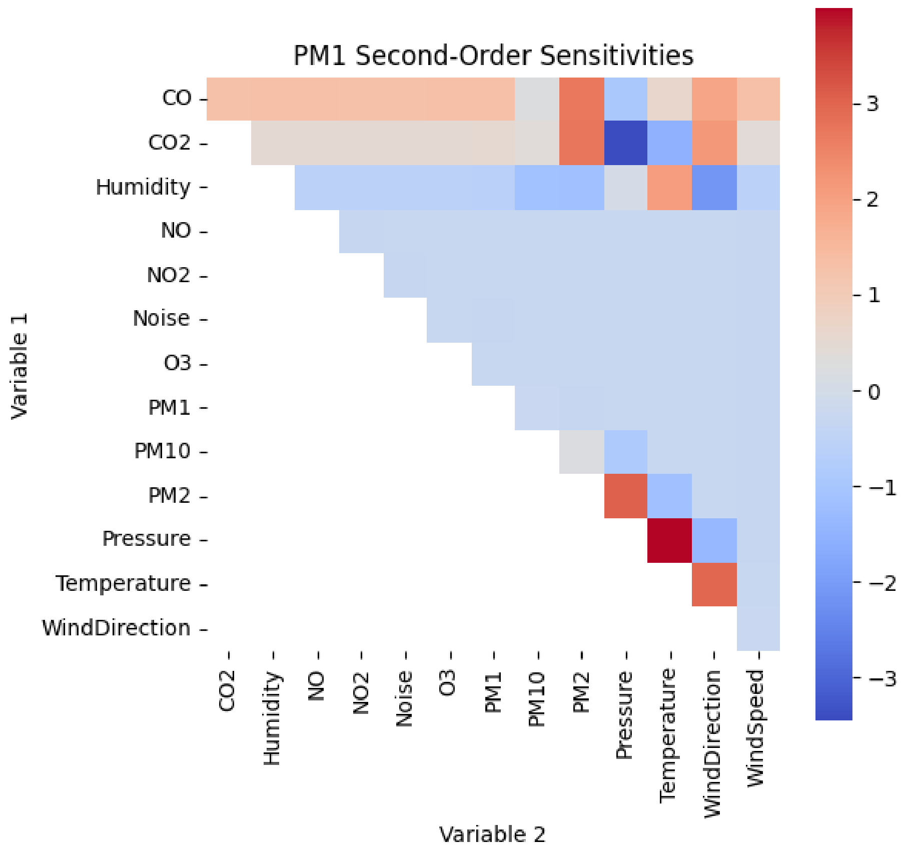
In general, the total sensitivities were much higher than the first-order sensitivities, suggesting that the model had developed higher-order relationships between the variables. It was also surprising to see a meteorological variable make up the largest total sensitivity for 5 out of the 8 air pollutants, suggesting that the incorporation of weather into the model was a significant addition to past forecasting methods in construction zones (see Table 5). A look at the results for a particular pollutant, PM_1, will be discussed. However, these findings can be similarly applied to the other air pollutants.

Figure xii: Total and First-Order Sensitivities of PM1



For PM_1, both pressure and temperature were the variables with the highest total-order sensitivities ('ST' column in Figure xii). This says that changes to the outside pressure or temperature consistently effect the predicted levels of PM_1 according to the model. These total sensitivity values are also significantly larger than their respective first-order sensitivities ('S1" column in Figure xii), suggesting that a majority of the effect on the total sensitivity is caused by higher-order sensitivities. This can be seen in the second-order sensitivity heatmap for PM_1, where the three pairs of variables with the largest sensitivities are $PM_{2.5}$ & Pressure, Pressure & Temperature, and Temperature & Wind Direction (see Figure xiii). This tells us that changes to both variables in these pairs consistently impact the predicted levels of PM_1, and that there exists higher-order relationships between the variables.

Figure xiii: Second-Order Sensitivities of PM1



An important note about sensitivity analysis is that the results are specific to the model used in the analysis. If the model is not an accurate representation of the real-world scenario, then the sensitivity analysis results are not significant. In this project's case, the best performing model was able to predict air pollutant levels 1-day ahead with an accuracy of over 80% (see Section 4.4), but this dropped significantly for longer-range forecasts. This suggests that there is some truth in the sensitivity analysis results, but it most likely is specific to daily changes in the pollutant levels. Another important consideration is how the device calculates the measurements, and whether there exists known interferences of specific sensors. For instance, it is known that humidity can interfere with the CO_2 sensor on the AirSENCE device, so the result that humidity had the largest total sensitivity for predicting CO_2 in Table 5 is likely "not real". However, the rest of the top sensitivities in Table 5 have not been seen to influence their resulting pollutant’s sensor, so those results are likely “real” and reveal important relationships that cause daily changes in pollutant levels.

4.6 Future Work

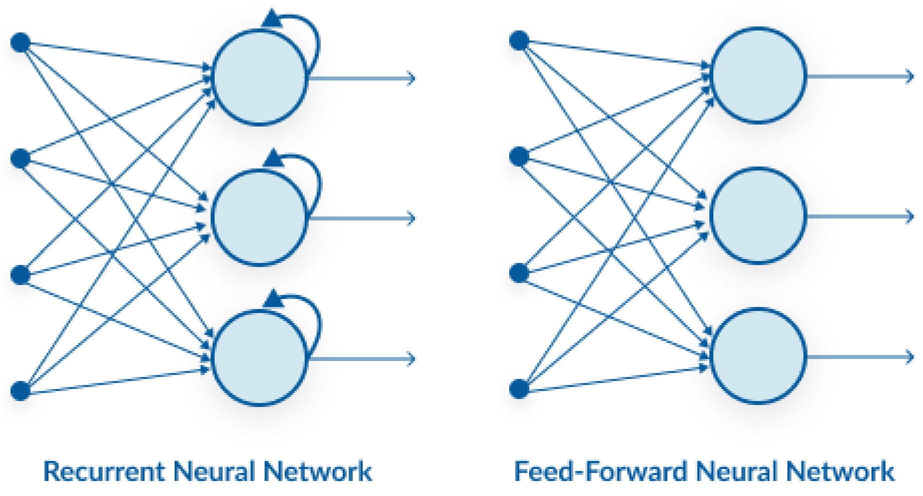
The original goal with this thesis was to allow construction companies to plan their working schedule in order to minimize the amount of air pollutants their workers and local residents would be exposed to. While the final model could accurately predict pollutant levels 1-day ahead, it was inaccurate for longer-range forecasts. Planning a work schedule with 1 day notice is not practical, and so further work must be done to improve the range of the model’s forecasting abilities.

One idea is to reduce the number of dimensions of the model. Currently, the problem is set up as a 14-variable multivariate time series. This is a very complex system, especially for a VARMA model, where each new variable exponentially increases the number of coefficients in the equation. Because of this, model training was a heavy computational task and limited the maximum training dataset size to 1000. If the number of parameters needed for the equation is reduced, model fitting will be more efficient and allow for larger training dataset sizes. A wider range of data would provide the model with examples of monthly, seasonal, or even yearly patterns found in air pollutants, improving the accuracy of the model.

One method to reduce the number of variables is through cointegration. If two variables have a long-run equilibrium, move together in such a way that their linear combination results in a stationary time series, and they share an underlying common stochastic trend, then they can be linearly combined to form a single variable without losing information about the overall system [32]. Cointegration can be determined through cointegration tests, for example the *Johansen Trace Test*. This can be achieved using the *statsmodels.VECM.coint_johansen* Python function [33].

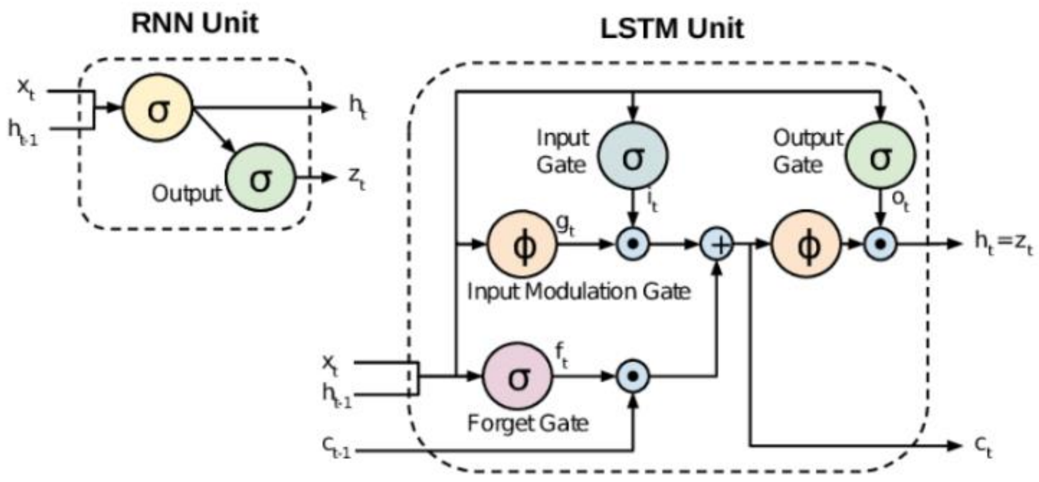
It would also be interesting to try using a neural network to model the system, instead of classic statistical models. Neural Networks have achieved success in multivariate time series forecasting tasks, particularly due to its ability to understand complex interdependencies between variables [34]. Specifically, a Long Short-Term Memory (LSTM) network should be used, which is a more sophisticated version of a Recurrent Neural Network (RNN). RNN's are considered the "standard" neural network for handling time-series data, due to its use of a feedback loop, where it uses prior outputs as inputs to the system.

Figure xiv: RNNs use a Feedback Loop [35]



A LSTM differs from an RNN, in that it features a "memory cell" that can remember information for longer periods time. This lets the network learn longer-term dependencies, such as weekly or seasonal patterns found in air pollutants and weather. LSTMs also deal with the vanishing and exploding gradient problem introduced by RNNs, through its inclusion of new "input" and "forget" gates (see Figure xv).

Figure xv: A LSTM Unit is a more Sophisticated RNN Unit [36]



A downside to using a neural network is a lack of transparency within model. Compared to a VARMA model, where the equation coefficients can be viewed and easily interpreted, the weights in the layers of a neural network are seemingly random. However, the sensitivity analysis performed in this paper can be similarly done using a trained neural network model, since it only looks at the output of the model from specifically chosen inputs. The impulse responses on the air pollutants can be observed, providing useful insight to an otherwise uninterpretable model.

5 Conclusion

In an attempt to better understand the relationship between meteorological variables and air pollutants near construction zones, statistical methods and sensitivity analysis techniques were applied to the multivariate time series problem. It was found that the incorporation of a moving average component with a vector autoregressive model (VARMA) could predict future pollutant levels more accurately than just the vector autoregressive model (VAR) alone. It was also observed that the size of the training data used had an impact on model performance. The training datasets that featured more data points covered a wider time period, allowing the model to better identify diurnal and longer-range patterns commonly seen in air pollutants and weather. This led to more accurate forecasting compared to models fitted using a smaller training data size. Lastly, due to the strict limits on the amount of training data points that can used computationally, the lowest resolution training data (60-minute) produced a more accurate model than the higher resolution data (15-minute, 1-minute). The maximum training dataset size of 1000 points meant the 60-minute dataset covered a much wider period (40 days) than the 15-minute and 1-minute datasets (10 days and 17 hours, respectively), allowing for examples of possible weekly or biweekly patterns to be observed by the model.

These findings were applied to create an optimal model, which predicted future air pollutant levels 1-day ahead with over 80% accuracy. However, prediction accuracy dropped significantly for 1-week and 2-week forecast lengths, likely due to the complexity of a 14 variable multivariate time series system and inherit randomness found in weather and air pollutant variables. Sobol sensitivity analysis was performed on this model, highlighting the existence of higher-order relationships between the variables. It also showed that 5 out of the 8 air pollutants had a meteorological variable as their top total sensitivity, reinforcing the decision to include weather variables in the forecasting of air pollutant levels.

While the optimal model could accurately predict air pollutant levels 1-day ahead, the original goal of using this model to schedule construction activity in order to minimize workers' exposure to air pollution was not achieved. This would realistically require a model that can accurately predict air pollutant levels weeks-ahead for it to be practical from a scheduling standpoint. However, the work done here represents a good starting-point for tackling the difficult task of long-term forecasting. The sensitivity analysis results showcasing interdependencies between weather and air pollutant variables can be further investigated. These can also be used to reduce the dimensions in the multivariate data, by creating individual models for each air pollutant using only the variables with the top total sensitivities. This would reduce the complexity of the model, allowing for a longer training data period, which has been proven to increase model accuracy. Steps to incorporate a neural network approach were also given, and the sensitivity analysis techniques highlighted in this paper can then be applied similarly.

6 References

[1] *Evans Research Group*. [Online]. Available: https://www.labs.chem-eng.utoronto.ca/evans/research/linking-emissions-and-health/. [Accessed: 22-Oct-2022].

[2] J. Teller, "Regulating urban densification: What factors should be used?," *Buildings and Cities*, vol. 2, no. 1, pp. 302–317, 2021.

[3] M. Draaisma, "Ontario premier says construction underway on New Toronto subway line | CBC news," *CBCnews*, 27-Mar-2022. [Online]. Available: https://www.cbc.ca/news/canada/toronto/ontario-line-official-breaking-ceremony-toronto-doug-ford-john-tory-1.6399282#. [Accessed: 22-Oct-2022].

[4] H. Yan, G. Ding, H. Li, Y. Wang, L. Zhang, Q. Shen, and K. Feng, "Field evaluation of the dust impacts from construction sites on surrounding areas: A city case study in China," *Sustainability*, vol. 11, no. 7, p. 1906, 2019.

[5] "Health consequences of air pollution on populations," *World Health Organization*, 15-Nov-2019. [Online]. Available: https://www.who.int/news/item/15-11-2019-what-are-health-consequences-of-air-pollution-on-populations. [Accessed: 24-Oct-2022].

[6] "Technical assistance document for the reporting of Daily Air ... - airnow," *AirNow*, 2018. [Online]. Available: https://www.airnow.gov/sites/default/files/2020-05/aqi-technical-assistance-document-sept2018.pdf. [Accessed: 24-Oct-2022].

[7] J. S. Evans and D. W. Cooper, "An inventory of particulate emissions from open sources," *Journal of the Air Pollution Control Association*, vol. 30, no. 12, pp. 1298–1303, 1980.

[8] I. A. Bergdahl, K. Torén, K. Eriksson, U. Hedlund, T. Nilsson, R. Flodin, and B. Järvholm, "Increased mortality in COPD among construction workers exposed to inorganic dust," *European Respiratory Journal*, vol. 23, no. 3, pp. 402–406, 2004.

[9] Y.-M. Chang, T.-C. Chang, and W.-K. Chen, "An estimation on overall emission rate of fugitive dust emitted from road construction activity," *Environmental Engineering Science*, vol. 16, no. 5, pp. 375–388, 1999.

[10] A. Font, T. Baker, I. S. Mudway, E. Purdie, C. Dunster, and G. W. Fuller, "Degradation in urban air quality from construction activity and increased traffic arising from a road widening scheme," *Science of The Total Environment*, vol. 497-498, pp. 123–132, 2014.

[11] "Rules on air quality and pollution," *ontario.ca*, 2022. [Online]. Available: https://www.ontario.ca/page/rules-air-quality-and-pollution. [Accessed: 20-Jan-2023].

[12] "Construction & Air Pollution," *Toronto Environmental Alliance*. [Online]. Available: https://www.torontoenvironment.org/construction_pollution. [Accessed: 21-Jan-2023].

[13] H. Li, A. Cheshmehzangi, Z. Zhang, Z. Su, S. Pourroostaei Ardakani, M. Sedrez, and A. Dawodu, "The correlation analysis between air quality and construction sites: Evaluation in the urban environment during the COVID-19 pandemic," *Sustainability*, vol. 14, no. 12, p. 7075, 2022.

[14] G. C. Reinsel, "Chapter 2 - Vector ARMA Time Series Models and Forecasting," in *Elements of multivariate time series analysis*, New York: Springer, 1993, pp. 21–51.

[15] "Variance-based sensitivity analysis," *Wikipedia*, 24-Feb-2023. [Online]. Available: https://en.wikipedia.org/wiki/Variance-based_sensitivity_analysis. [Accessed: 08-Apr-2023].

[16] A. Saltelli, *Global sensitivity analysis: The Primer*. Chichester, England: John Wiley, 2008.

[17] D. M. Stieb, R. T. Burnett, M. Smith-Doiron, O. Brion, H. H. Shin, and V. Economou, "A new multipollutant, no-threshold air quality health index based on short-term associations observed in daily time-series analyses," *Journal of the Air & Waste Management Association*, vol. 58, no. 3, pp. 435–450, 2008.

[18] " 6.4.4.2. Stationarity," *Engineering Statistics Handbook*. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm. [Accessed: 16-Apr-2023].

[19] R. Rasheed, "Why does stationarity matter in time series analysis?," *Medium*, 12-Jul-2020. [Online]. Available: https://towardsdatascience.com/why-does-stationarity-matter-in-time-series-analysis-e2fb7be74454. [Accessed: 10-Apr-2023].

[20] S. Prabhakaran, "Augmented dickey-fuller (ADF) test - must read guide - ml+," *Machine Learning Plus*, 04-Apr-2022. [Online]. Available: https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/. [Accessed: 10-Apr-2023].

[21] "Statsmodels.tsa.stattools.adfuller," *statsmodels*. [Online]. Available: https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html. [Accessed: 02-Apr-2023].

[22] M. Fuchs, "Time series analysis - regression extension techniques for multivariate time series - Michael Fuchs Python," *MFuchs*, 29-Oct-2020. [Online]. Available: https://michael-fuchs-python.netlify.app/2020/10/29/time-series-analysis-regression-extension-techniques-for-forecasting-multivariate-variables/#stationarity. [Accessed: 04-Apr-2023].

[23] "Akaike information criterion," *Akaike Information Criterion - an overview | ScienceDirect Topics*. [Online]. Available:

https://www.sciencedirect.com/topics/social-sciences/akaike-information-criterion. [Accessed: 11-Apr-2023].

[24] "Statsmodels.tsa.vector_ar.Var_model.var.select_order," *statsmodels*. [Online]. Available: https://www.statsmodels.org/dev/generated/statsmodels.tsa.vector_ar.var_model.VAR.select_order.html. [Accessed: 10-Apr-2023].

[25] "Statsmodels.tsa.statespace.varmax.VARMAX," *statsmodels*. [Online]. Available: https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.varmax.VARMAX.html. [Accessed: 10-Apr-2023].

[26] "PMDARIMA.ARIMA.AUTO_ARIMA," *pmdarima.arima.auto_arima - pmdarima 2.0.3 documentation*. [Online]. Available: https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html. [Accessed: 11-Apr-2023].

[27] S. Glen, "Mean absolute percentage error (MAPE)," *Statistics How To*, 27-May-2022. [Online]. Available: https://www.statisticshowto.com/mean-absolute-percentage-error-mape/. [Accessed: 13-Apr-2023].

[28] "Sensitivity Analysis Library in python#," *SALib*. [Online]. Available: https://salib.readthedocs.io/en/latest/index.html. [Accessed: 12-Apr-2023].

[29] M. Zhang, "Time Series: Autoregressive models AR, MA, ARMA, ARIMA," *CS3750 Course Page*, 23-Oct-2018. [Online]. Available: https://people.cs.pitt.edu/~milos/courses/cs3750/lectures/class16.pdf. [Accessed: 03-Apr-2023].

[30] "Why is an arma model A parsimonous approximation of an AR model?," *Stack Exchange*, 01-Mar-2015. [Online]. Available: https://stats.stackexchange.com/questions/103368/why-is-an-arma-model-a-parsimonous-approximation-of-an-ar-model. [Accessed: 07-Apr-2023].

[31] S. Sukkhum, A. Lim, T. Ingviya, and R. Saelim, "Seasonal patterns and trends of air pollution in the upper northern Thailand from 2004 to 2018," *Aerosol and Air Quality Research*, vol. 22, no. 5, p. 210318, 2022.

[32] Eric, "A guide to conducting Cointegration tests," *Aptech*, 18-Apr-2022. [Online]. Available: https://www.aptech.com/blog/a-guide-to-conducting-cointegration-tests/. [Accessed: 01-Apr-2023].

[33] "Statsmodels.tsa.vector_ar.vecm.coint_johansen," *statsmodels*. [Online]. Available: https://www.statsmodels.org/dev/generated/statsmodels.tsa.vector_ar.vecm.coint_johansen.html. [Accessed: 02-Apr-2023].

[34] A. N. M. F. Faisal, A. Rahman, M. T. Habib, A. H. Siddique, M. Hasan, and M. M. Khan, "Neural networks based multivariate time series forecasting of solar radiation using meteorological data of different cities of Bangladesh," *Results in Engineering*, vol. 13, p. 100365, 2022.

[35] A. Pai, “Ann vs CNN vs RNN: Types of neural networks,” *Analytics Vidhya*, 19-Oct-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/. [Accessed: 15-Oct-2022].

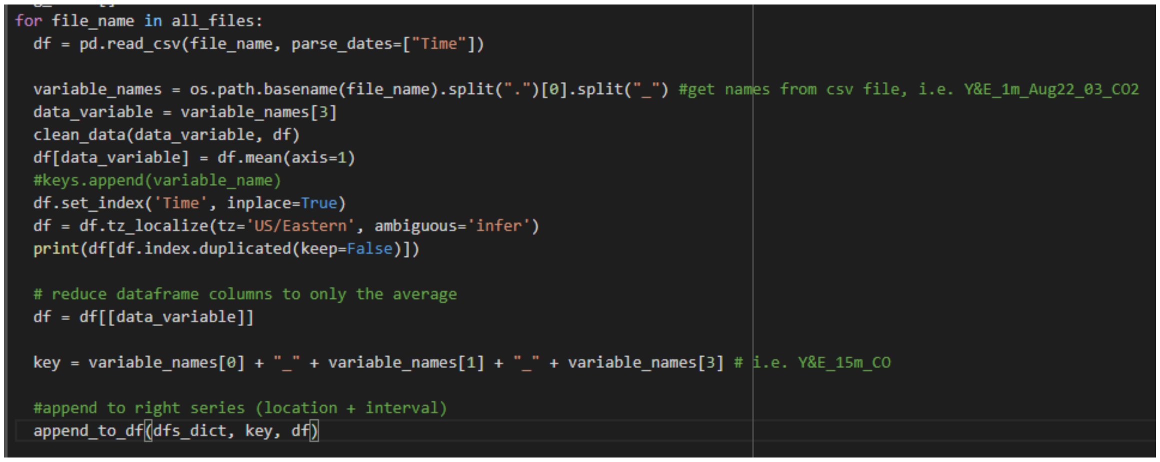
[36] A. Tripathi, “What is the main difference between RNN and LSTM: NLP: RNN VS LSTM,” *Data Science Duniya*, 18-Jul-2022. [Online]. Available: https://ashutoshtripathi.com/2021/07/02/what-is-the-main-difference-between-rnn-and-lstm-nlp-rnn-vs-lstm/. [Accessed: 16-Oct-2022].

Appendix

Full source code, figures, and saved models can be found at https://github.com/markuskunej/air-pollution-thesis

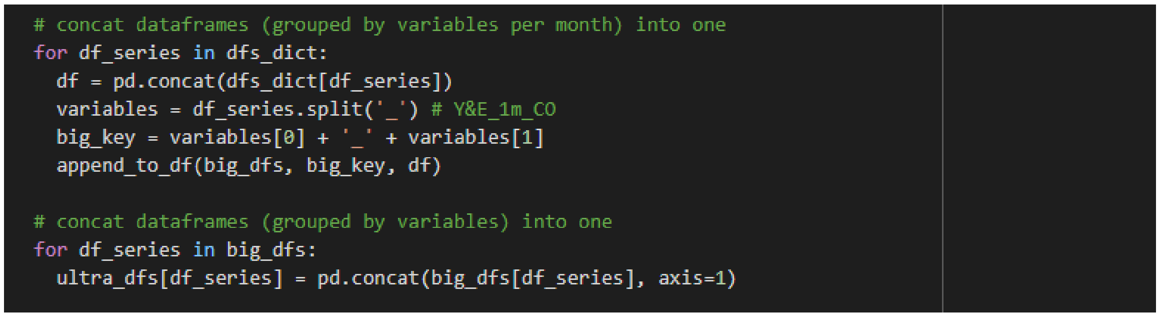
A. Key Methods from Project Codebase

A.1 Averaging Across 10 Devices



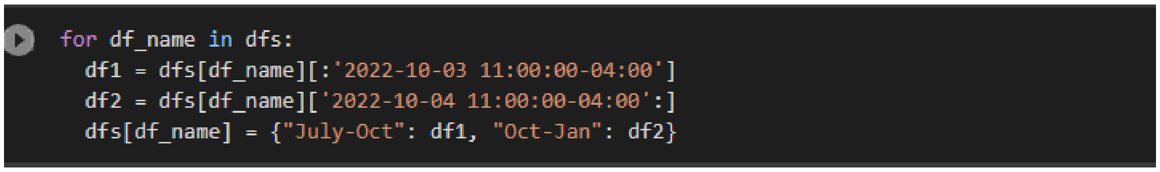
```python
for file_name in all_files:
  df = pd.read_csv(file_name, parse_dates=["Time"])

  variable_names = os.path.basename(file_name).split(".")[0].split("_") #get names from csv file, i.e. Y&E_1m_Aug22_03_CO2
  data_variable = variable_names[3]
  clean_data(data_variable, df)
  df[data_variable] = df.mean(axis=1)
  #keys.append(variable_name)
  df.set_index('Time', inplace=True)
  df = df.tz_localize(tz='US/Eastern', ambiguous='infer')
  print(df[df.index.duplicated(keep=False)])

  # reduce dataframe columns to only the average
  df = df[[data_variable]]

  key = variable_names[0] + "_" + variable_names[1] + "_" + variable_names[3] # i.e. Y&E_15m_CO

  #append to right series (location + interval)
  append_to_df(dfs_dict, key, df)
```

A.2 Concatenate Monthly and Variable DataFrames into One



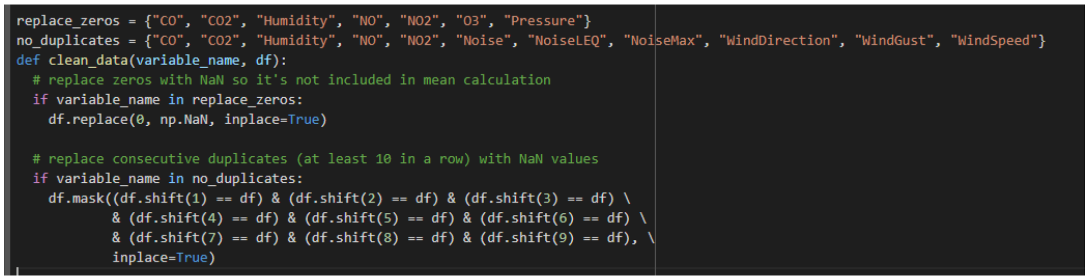
```python
# concat dataframes (grouped by variables per month) into one
for df_series in dfs_dict:
  df = pd.concat(dfs_dict[df_series])
  variables = df_series.split('_') # Y&E_1m_CO
  big_key = variables[0] + '_' + variables[1]
  append_to_df(big_dfs, big_key, df)

# concat dataframes (grouped by variables) into one
for df_series in big_dfs:
  ultra_dfs[df_series] = pd.concat(big_dfs[df_series], axis=1)
```

A.3 Remove Affected 24-hour period from Data



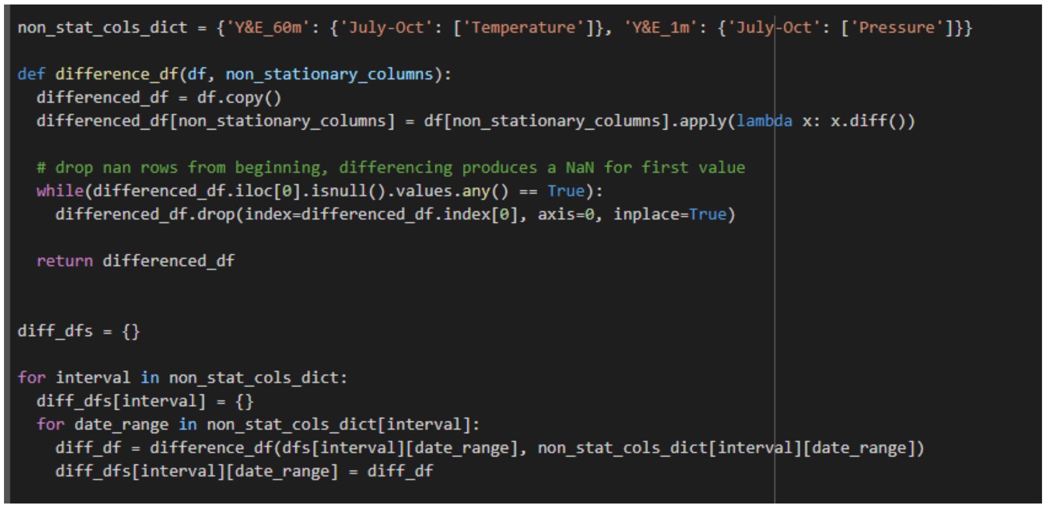
```python
for df_name in dfs:
  df1 = dfs[df_name][:'2022-10-03 11:00:00-04:00']
  df2 = dfs[df_name]['2022-10-04 11:00:00-04:00':]
  dfs[df_name] = {"July-Oct": df1, "Oct-Jan": df2}
```

A.4 Replace ‘0’s and Repeated Values with NaN



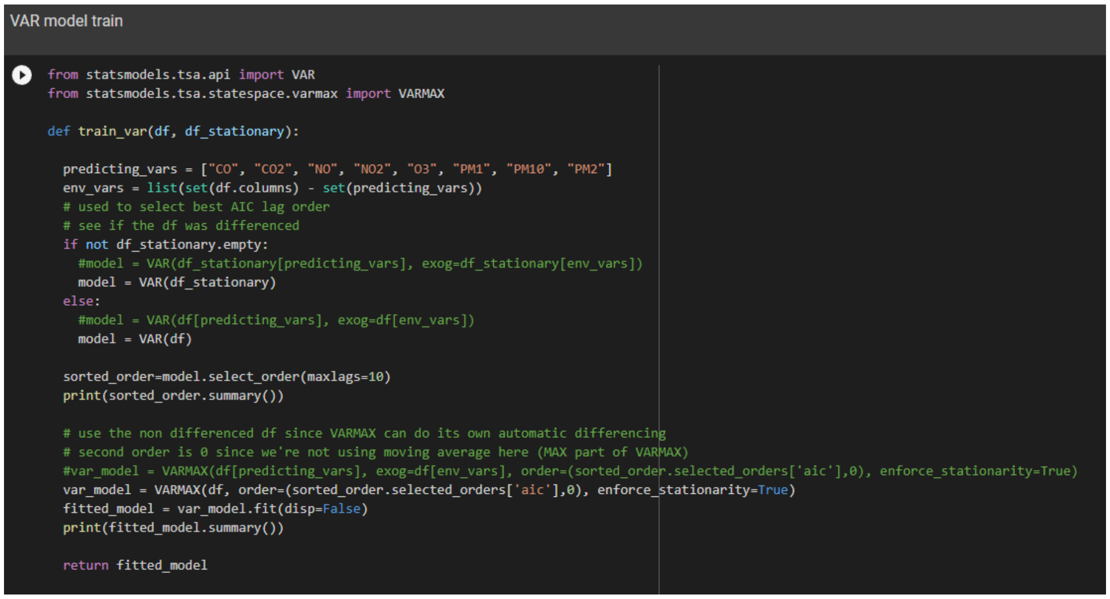
```
replace_zeros = {"CO", "CO2", "Humidity", "NO", "NO2", "O3", "Pressure"}
no_duplicates = {"CO", "CO2", "Humidity", "NO", "NO2", "Noise", "NoiseLEQ", "NoiseMax", "WindDirection", "WindGust", "WindSpeed"}
def clean_data(variable_name, df):
  # replace zeros with NaN so it's not included in mean calculation
  if variable_name in replace_zeros:
    df.replace(0, np.NaN, inplace=True)

  # replace consecutive duplicates (at least 10 in a row) with NaN values
  if variable_name in no_duplicates:
    df.mask((df.shift(1) == df) & (df.shift(2) == df) & (df.shift(3) == df) \
            & (df.shift(4) == df) & (df.shift(5) == df) & (df.shift(6) == df) \
            & (df.shift(7) == df) & (df.shift(8) == df) & (df.shift(9) == df), \
            inplace=True)
```

A.5 Difference Non-Stationary Variables



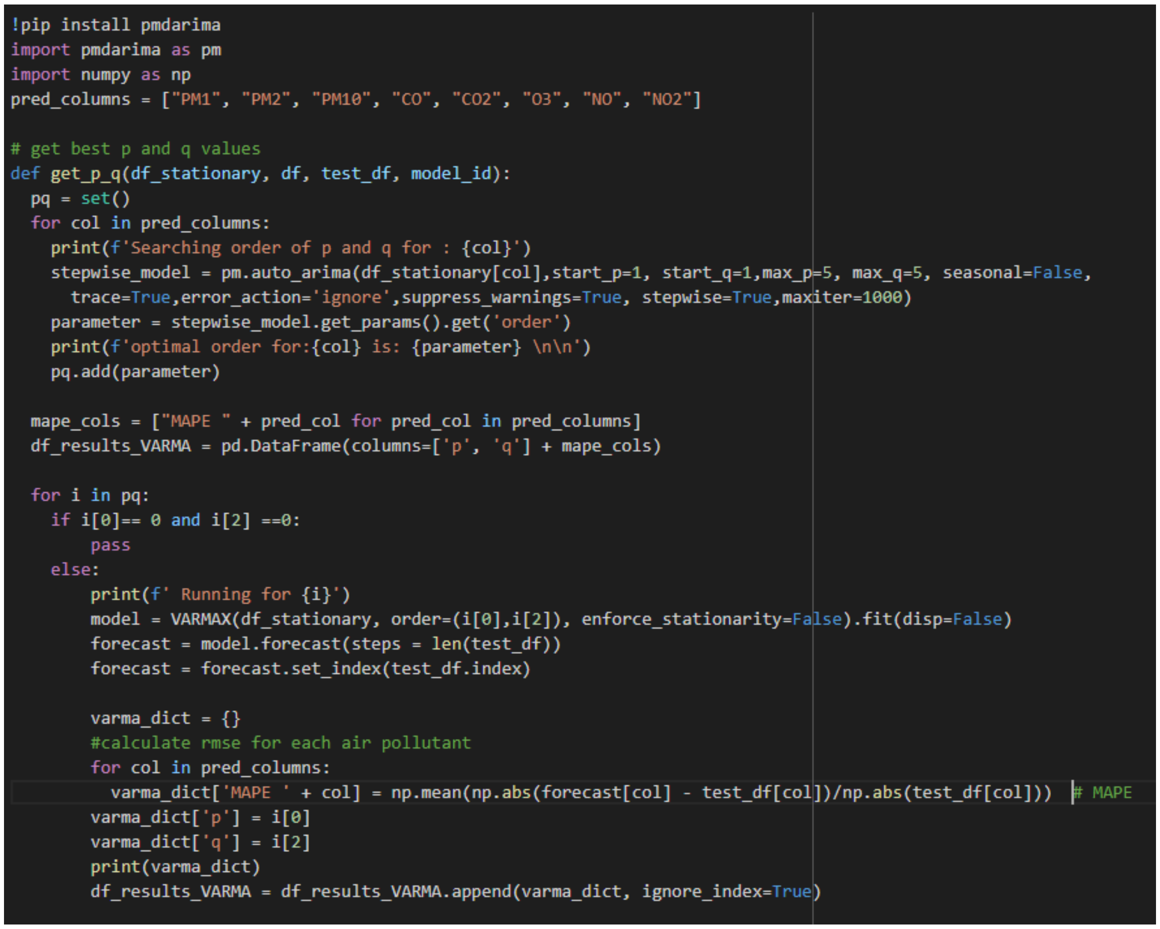
```
non_stat_cols_dict = {'Y&E_60m': {'July-Oct': ['Temperature']}, 'Y&E_1m': {'July-Oct': ['Pressure']}}

def difference_df(df, non_stationary_columns):
  differenced_df = df.copy()
  differenced_df[non_stationary_columns] = df[non_stationary_columns].apply(lambda x: x.diff())

  # drop nan rows from beginning, differencing produces a NaN for first value
  while(differenced_df.iloc[0].isnull().values.any() == True):
    differenced_df.drop(index=differenced_df.index[0], axis=0, inplace=True)

  return differenced_df


diff_dfs = {}

for interval in non_stat_cols_dict:
  diff_dfs[interval] = {}
  for date_range in non_stat_cols_dict[interval]:
    diff_df = difference_df(dfs[interval][date_range], non_stat_cols_dict[interval][date_range])
    diff_dfs[interval][date_range] = diff_df
```

A.6 VAR Lag Order Selection and Model Training

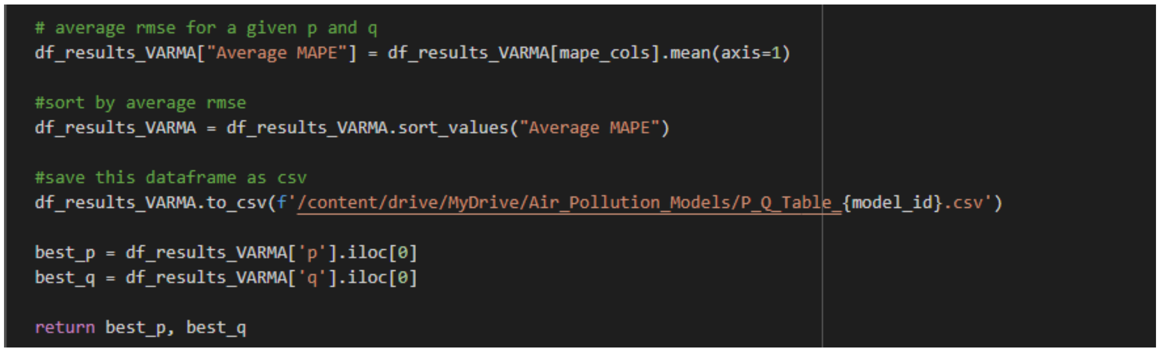


```
from statsmodels.tsa.api import VAR
from statsmodels.tsa.statespace.varmax import VARMAX

def train_var(df, df_stationary):

  predicting_vars = ["CO", "CO2", "NO", "NO2", "O3", "PM1", "PM10", "PM2"]
  env_vars = list(set(df.columns) - set(predicting_vars))
  # used to select best AIC lag order
  # see if the df was differenced
  if not df_stationary.empty:
    #model = VAR(df_stationary[predicting_vars], exog=df_stationary[env_vars])
    model = VAR(df_stationary)
  else:
    #model = VAR(df[predicting_vars], exog=df[env_vars])
    model = VAR(df)

  sorted_order=model.select_order(maxlags=10)
  print(sorted_order.summary())

  # use the non differenced df since VARMAX can do its own automatic differencing
  # second order is 0 since we're not using moving average here (MAX part of VARMAX)
  #var_model = VARMAX(df[predicting_vars], exog=df[env_vars], order=(sorted_order.selected_orders['aic'],0), enforce_stationarity=True)
  var_model = VARMAX(df, order=(sorted_order.selected_orders['aic'],0), enforce_stationarity=True)
  fitted_model = var_model.fit(disp=False)
  print(fitted_model.summary())

  return fitted_model
```

A.7 Get Best p and q Order for VARMA Training

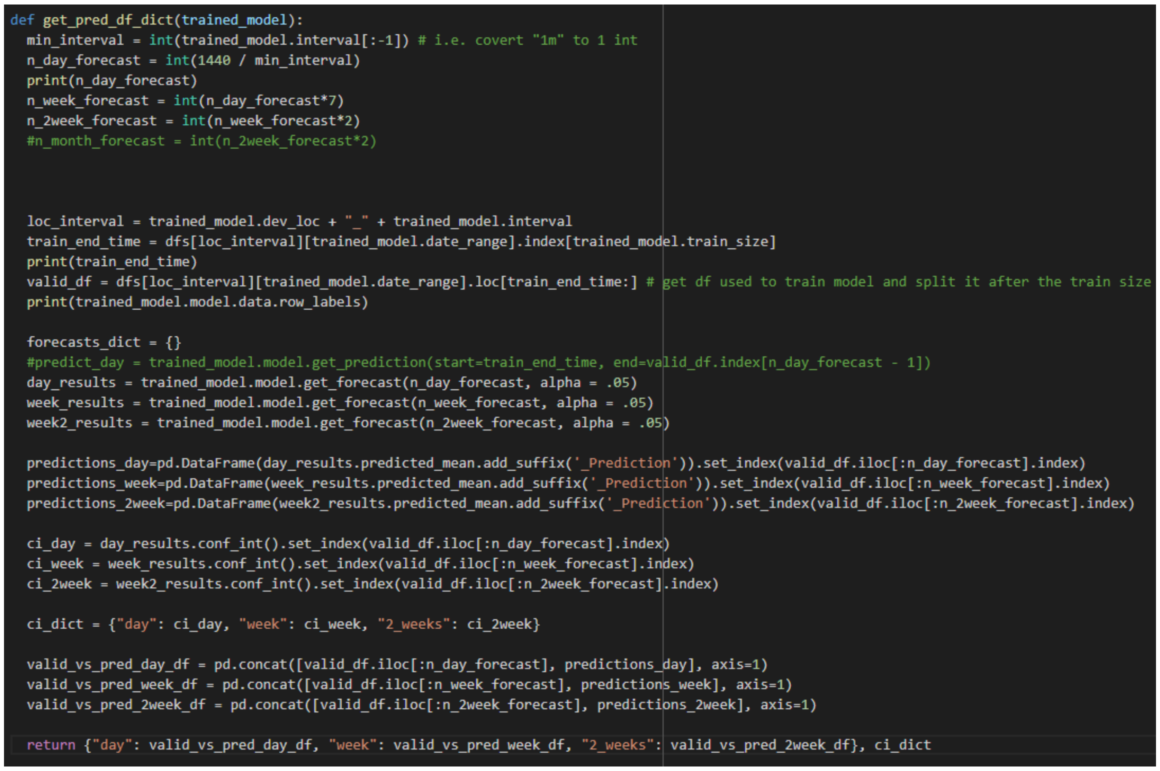


```
!pip install pmdarima
import pmdarima as pm
import numpy as np
pred_columns = ["PM1", "PM2", "PM10", "CO", "CO2", "O3", "NO", "NO2"]

# get best p and q values
def get_p_q(df_stationary, df, test_df, model_id):
  pq = set()
  for col in pred_columns:
    print(f'Searching order of p and q for : {col}')
    stepwise_model = pm.auto_arima(df_stationary[col],start_p=1, start_q=1,max_p=5, max_q=5, seasonal=False,
      trace=True,error_action='ignore',suppress_warnings=True, stepwise=True,maxiter=1000)
    parameter = stepwise_model.get_params().get('order')
    print(f'optimal order for:{col} is: {parameter} \n\n')
    pq.add(parameter)

  mape_cols = ["MAPE " + pred_col for pred_col in pred_columns]
  df_results_VARMA = pd.DataFrame(columns=['p', 'q'] + mape_cols)

  for i in pq:
    if i[0]== 0 and i[2] ==0:
        pass
    else:
        print(f' Running for {i}')
        model = VARMAX(df_stationary, order=(i[0],i[2]), enforce_stationarity=False).fit(disp=False)
        forecast = model.forecast(steps = len(test_df))
        forecast = forecast.set_index(test_df.index)

        varma_dict = {}
        #calculate rmse for each air pollutant
        for col in pred_columns:
          varma_dict['MAPE ' + col] = np.mean(np.abs(forecast[col] - test_df[col])/np.abs(test_df[col]))  # MAPE
        varma_dict['p'] = i[0]
        varma_dict['q'] = i[2]
        print(varma_dict)
        df_results_VARMA = df_results_VARMA.append(varma_dict, ignore_index=True)
```



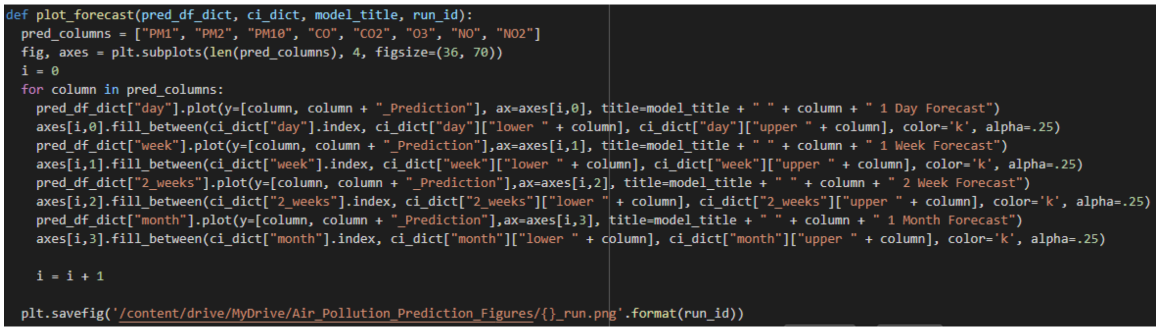
```
  # average rmse for a given p and q
  df_results_VARMA["Average MAPE"] = df_results_VARMA[mape_cols].mean(axis=1)

  #sort by average rmse
  df_results_VARMA = df_results_VARMA.sort_values("Average MAPE")

  #save this dataframe as csv
  df_results_VARMA.to_csv(f'/content/drive/MyDrive/Air_Pollution_Models/P_Q_Table_{model_id}.csv')

  best_p = df_results_VARMA['p'].iloc[0]
  best_q = df_results_VARMA['q'].iloc[0]

  return best_p, best_q
```

A.8 Make 1-day, 1-week, and 2-week Forecasts for a Trained Model



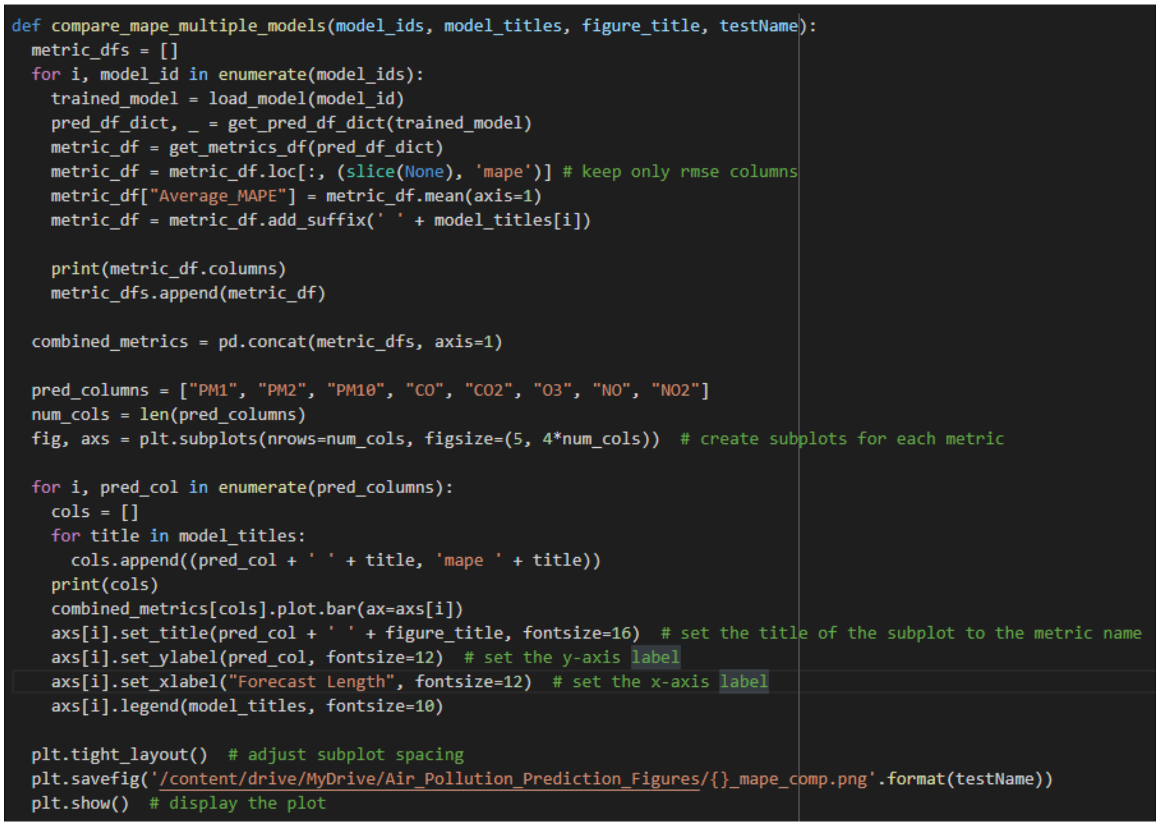
```
def get_pred_df_dict(trained_model):
  min_interval = int(trained_model.interval[:-1]) # i.e. covert "1m" to 1 int
  n_day_forecast = int(1440 / min_interval)
  print(n_day_forecast)
  n_week_forecast = int(n_day_forecast*7)
  n_2week_forecast = int(n_week_forecast*2)
  #n_month_forecast = int(n_2week_forecast*2)



  loc_interval = trained_model.dev_loc + "_" + trained_model.interval
  train_end_time = dfs[loc_interval][trained_model.date_range].index[trained_model.train_size]
  print(train_end_time)
  valid_df = dfs[loc_interval][trained_model.date_range].loc[train_end_time:] # get df used to train model and split it after the train size
  print(trained_model.model.data.row_labels)

  forecasts_dict = {}
  #predict_day = trained_model.model.get_prediction(start=train_end_time, end=valid_df.index[n_day_forecast - 1])
  day_results = trained_model.model.get_forecast(n_day_forecast, alpha = .05)
  week_results = trained_model.model.get_forecast(n_week_forecast, alpha = .05)
  week2_results = trained_model.model.get_forecast(n_2week_forecast, alpha = .05)

  predictions_day=pd.DataFrame(day_results.predicted_mean.add_suffix('_Prediction')).set_index(valid_df.iloc[:n_day_forecast].index)
  predictions_week=pd.DataFrame(week_results.predicted_mean.add_suffix('_Prediction')).set_index(valid_df.iloc[:n_week_forecast].index)
  predictions_2week=pd.DataFrame(week2_results.predicted_mean.add_suffix('_Prediction')).set_index(valid_df.iloc[:n_2week_forecast].index)

  ci_day = day_results.conf_int().set_index(valid_df.iloc[:n_day_forecast].index)
  ci_week = week_results.conf_int().set_index(valid_df.iloc[:n_week_forecast].index)
  ci_2week = week2_results.conf_int().set_index(valid_df.iloc[:n_2week_forecast].index)

  ci_dict = {"day": ci_day, "week": ci_week, "2_weeks": ci_2week}

  valid_vs_pred_day_df = pd.concat([valid_df.iloc[:n_day_forecast], predictions_day], axis=1)
  valid_vs_pred_week_df = pd.concat([valid_df.iloc[:n_week_forecast], predictions_week], axis=1)
  valid_vs_pred_2week_df = pd.concat([valid_df.iloc[:n_2week_forecast], predictions_2week], axis=1)

  return {"day": valid_vs_pred_day_df, "week": valid_vs_pred_week_df, "2_weeks": valid_vs_pred_2week_df}, ci_dict
```

A.9 Using matplotlib to Plot Forecasts and the 95% Confidence Interval

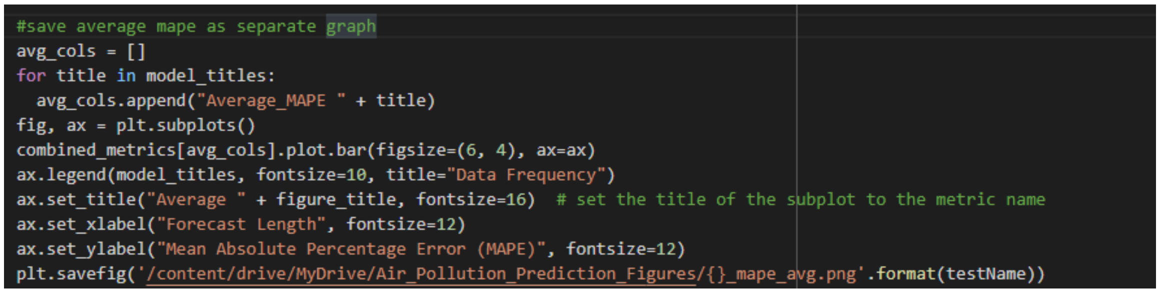


```
def plot_forecast(pred_df_dict, ci_dict, model_title, run_id):
  pred_columns = ["PM1", "PM2", "PM10", "CO", "CO2", "O3", "NO", "NO2"]
  fig, axes = plt.subplots(len(pred_columns), 4, figsize=(36, 70))
  i = 0
  for column in pred_columns:
    pred_df_dict["day"].plot(y=[column, column + "_Prediction"], ax=axes[i,0], title=model_title + " " + column + " 1 Day Forecast")
    axes[i,0].fill_between(ci_dict["day"].index, ci_dict["day"]["lower " + column], ci_dict["day"]["upper " + column], color='k', alpha=.25)
    pred_df_dict["week"].plot(y=[column, column + "_Prediction"],ax=axes[i,1], title=model_title + " " + column + " 1 Week Forecast")
    axes[i,1].fill_between(ci_dict["week"].index, ci_dict["week"]["lower " + column], ci_dict["week"]["upper " + column], color='k', alpha=.25)
    pred_df_dict["2_weeks"].plot(y=[column, column + "_Prediction"],ax=axes[i,2], title=model_title + " " + column + " 2 Week Forecast")
    axes[i,2].fill_between(ci_dict["2_weeks"].index, ci_dict["2_weeks"]["lower " + column], ci_dict["2_weeks"]["upper " + column], color='k', alpha=.25)
    pred_df_dict["month"].plot(y=[column, column + "_Prediction"],ax=axes[i,3], title=model_title + " " + column + " 1 Month Forecast")
    axes[i,3].fill_between(ci_dict["month"].index, ci_dict["month"]["lower " + column], ci_dict["month"]["upper " + column], color='k', alpha=.25)

    i = i + 1

  plt.savefig('/content/drive/MyDrive/Air_Pollution_Prediction_Figures/{}_run.png'.format(run_id))
```

A.10 Compare a Group of Models Using MAPE

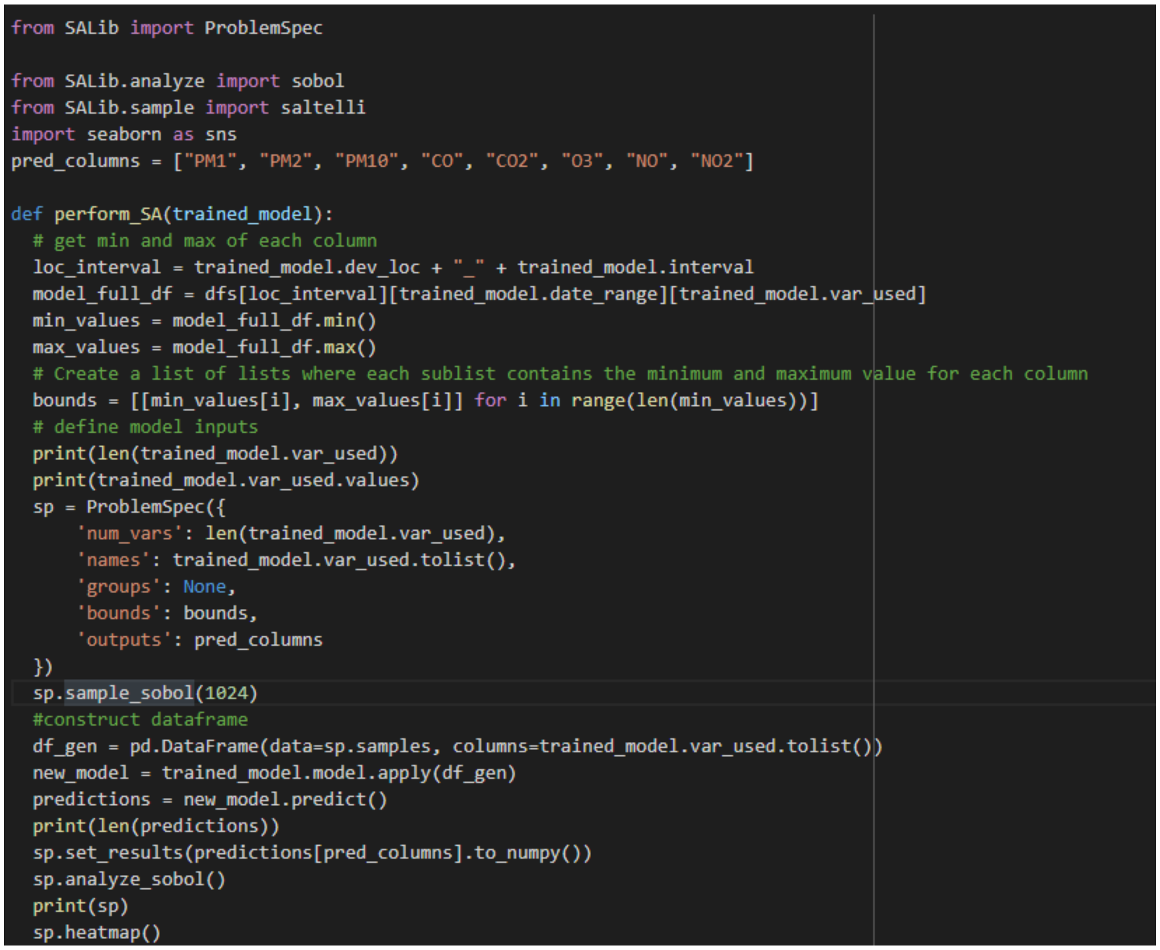


```python
def compare_mape_multiple_models(model_ids, model_titles, figure_title, testName):
  metric_dfs = []
  for i, model_id in enumerate(model_ids):
    trained_model = load_model(model_id)
    pred_df_dict, _ = get_pred_df_dict(trained_model)
    metric_df = get_metrics_df(pred_df_dict)
    metric_df = metric_df.loc[:, (slice(None), 'mape')] # keep only rmse columns
    metric_df["Average_MAPE"] = metric_df.mean(axis=1)
    metric_df = metric_df.add_suffix(' ' + model_titles[i])

    print(metric_df.columns)
    metric_dfs.append(metric_df)

  combined_metrics = pd.concat(metric_dfs, axis=1)

  pred_columns = ["PM1", "PM2", "PM10", "CO", "CO2", "O3", "NO", "NO2"]
  num_cols = len(pred_columns)
  fig, axs = plt.subplots(nrows=num_cols, figsize=(5, 4*num_cols))  # create subplots for each metric

  for i, pred_col in enumerate(pred_columns):
    cols = []
    for title in model_titles:
      cols.append((pred_col + ' ' + title, 'mape ' + title))
    print(cols)
    combined_metrics[cols].plot.bar(ax=axs[i])
    axs[i].set_title(pred_col + ' ' + figure_title, fontsize=16)  # set the title of the subplot to the metric name
    axs[i].set_ylabel(pred_col, fontsize=12)  # set the y-axis label
    axs[i].set_xlabel("Forecast Length", fontsize=12)  # set the x-axis label
    axs[i].legend(model_titles, fontsize=10)

  plt.tight_layout()  # adjust subplot spacing
  plt.savefig('/content/drive/MyDrive/Air_Pollution_Prediction_Figures/{}_mape_comp.png'.format(testName))
  plt.show()  # display the plot
```

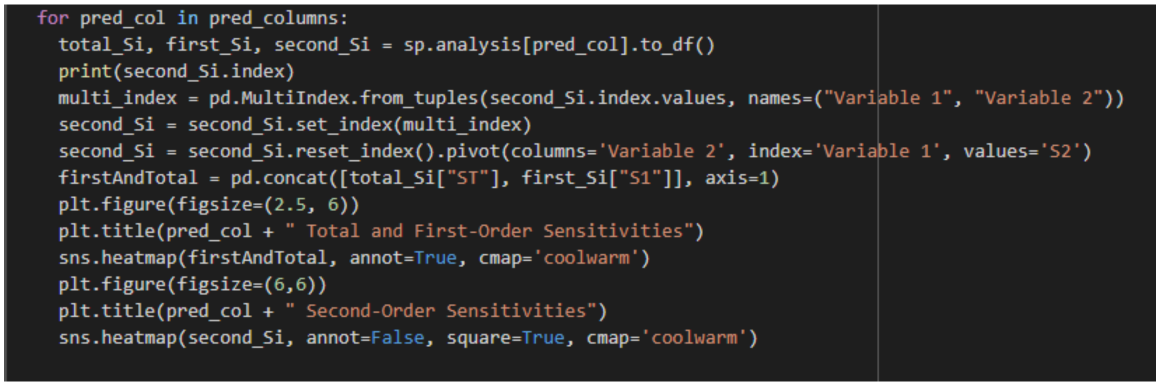


```python
#save average mape as separate graph
avg_cols = []
for title in model_titles:
  avg_cols.append("Average_MAPE " + title)
fig, ax = plt.subplots()
combined_metrics[avg_cols].plot.bar(figsize=(6, 4), ax=ax)
ax.legend(model_titles, fontsize=10, title="Data Frequency")
ax.set_title("Average " + figure_title, fontsize=16)  # set the title of the subplot to the metric name
ax.set_xlabel("Forecast Length", fontsize=12)
ax.set_ylabel("Mean Absolute Percentage Error (MAPE)", fontsize=12)
plt.savefig('/content/drive/MyDrive/Air_Pollution_Prediction_Figures/{}_mape_avg.png'.format(testName))
```

A.11 Saltelli Sequence Generation and Sobol Sensitivity Analysis using SALib



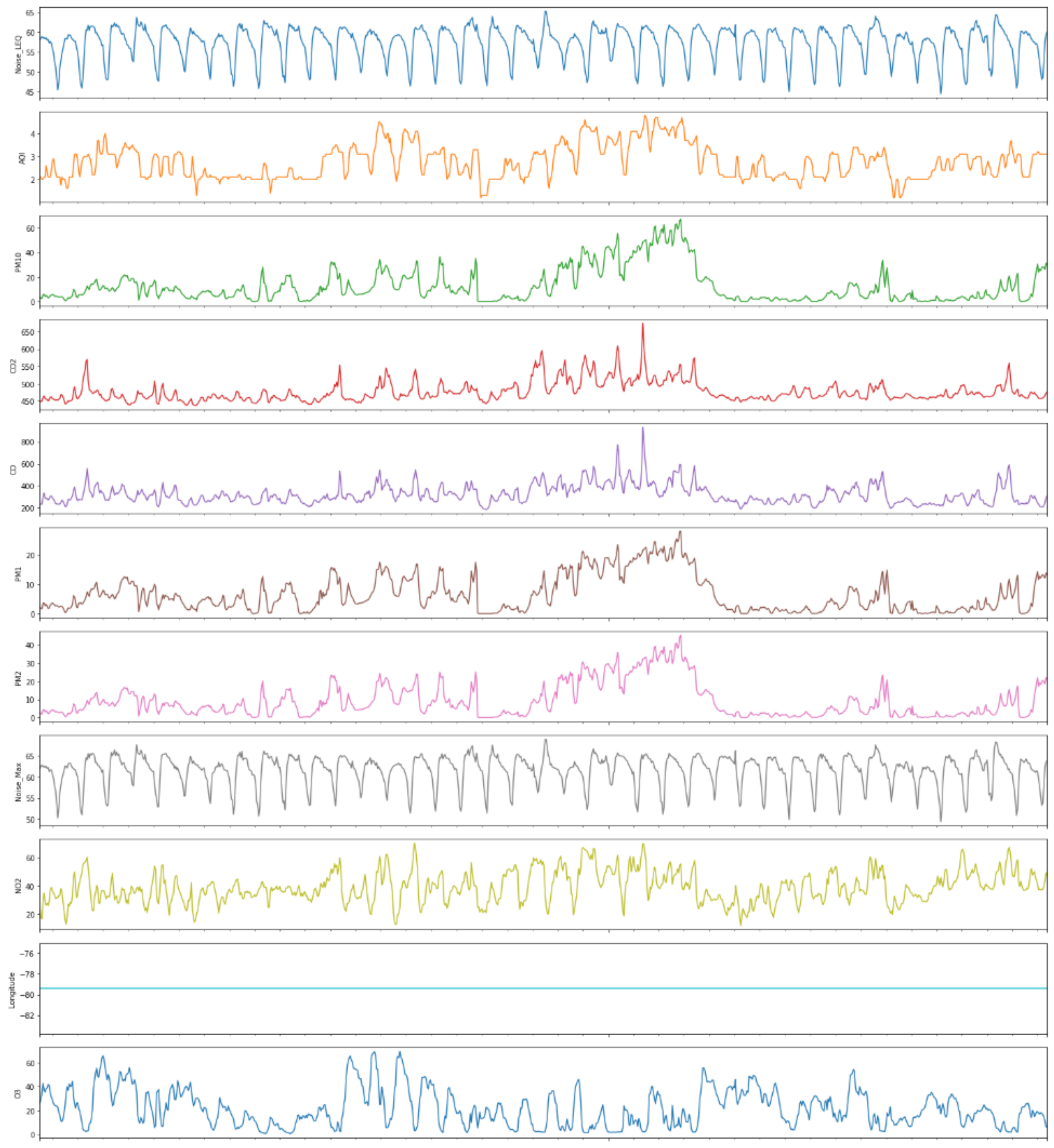
```python
from SALib import ProblemSpec

from SALib.analyze import sobol
from SALib.sample import saltelli
import seaborn as sns
pred_columns = ["PM1", "PM2", "PM10", "CO", "CO2", "O3", "NO", "NO2"]

def perform_SA(trained_model):
  # get min and max of each column
  loc_interval = trained_model.dev_loc + "_" + trained_model.interval
  model_full_df = dfs[loc_interval][trained_model.date_range][trained_model.var_used]
  min_values = model_full_df.min()
  max_values = model_full_df.max()
  # Create a list of lists where each sublist contains the minimum and maximum value for each column
  bounds = [[min_values[i], max_values[i]] for i in range(len(min_values))]
  # define model inputs
  print(len(trained_model.var_used))
  print(trained_model.var_used.values)
  sp = ProblemSpec({
      'num_vars': len(trained_model.var_used),
      'names': trained_model.var_used.tolist(),
      'groups': None,
      'bounds': bounds,
      'outputs': pred_columns
  })
  sp.sample_sobol(1024)
  #construct dataframe
  df_gen = pd.DataFrame(data=sp.samples, columns=trained_model.var_used.tolist())
  new_model = trained_model.model.apply(df_gen)
  predictions = new_model.predict()
  print(len(predictions))
  sp.set_results(predictions[pred_columns].to_numpy())
  sp.analyze_sobol()
  print(sp)
  sp.heatmap()
```

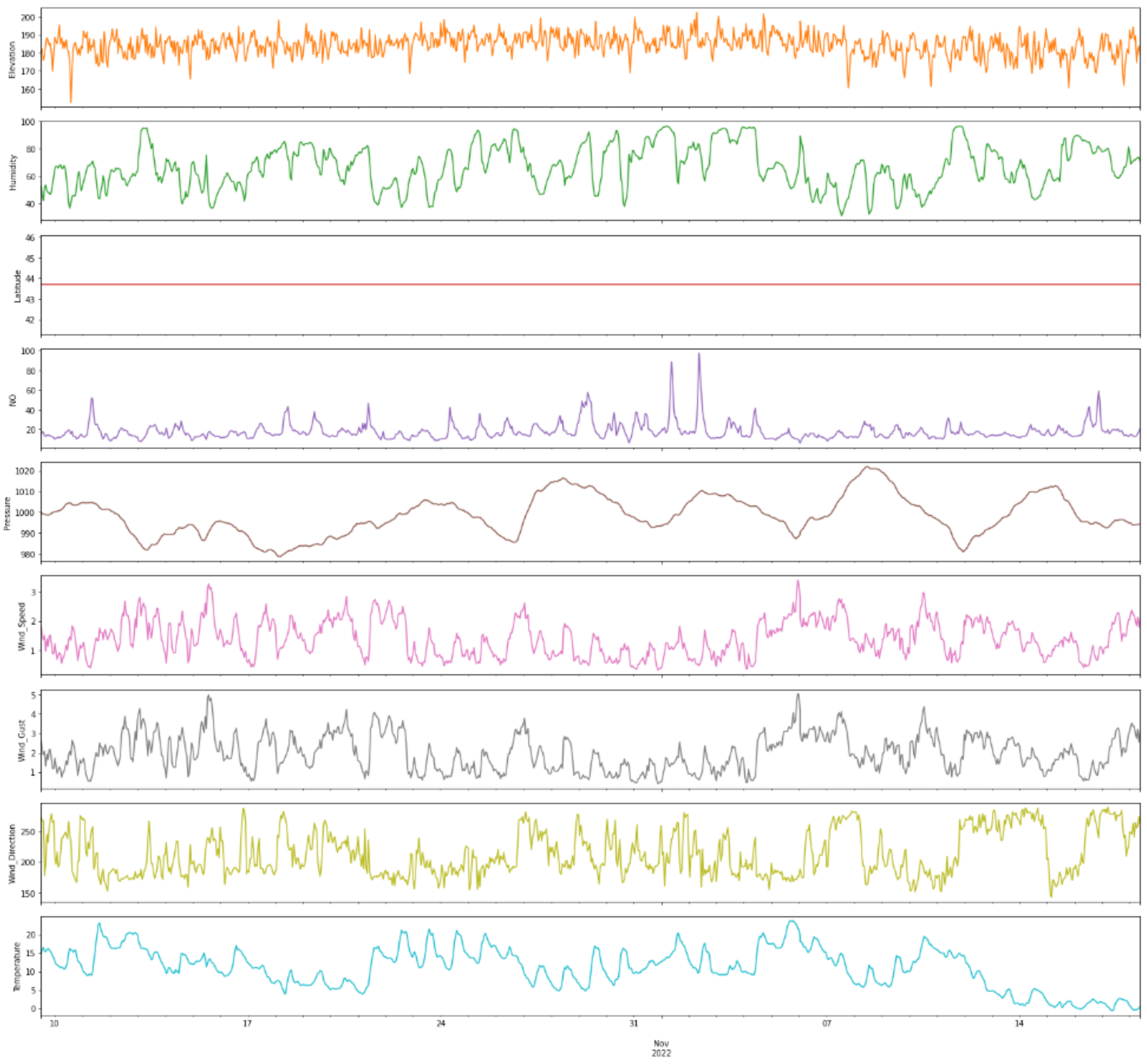


```python
  for pred_col in pred_columns:
    total_Si, first_Si, second_Si = sp.analysis[pred_col].to_df()
    print(second_Si.index)
    multi_index = pd.MultiIndex.from_tuples(second_Si.index.values, names=("Variable 1", "Variable 2"))
    second_Si = second_Si.set_index(multi_index)
    second_Si = second_Si.reset_index().pivot(columns='Variable 2', index='Variable 1', values='S2')
    firstAndTotal = pd.concat([total_Si["ST"], first_Si["S1"]], axis=1)
    plt.figure(figsize=(2.5, 6))
    plt.title(pred_col + " Total and First-Order Sensitivities")
    sns.heatmap(firstAndTotal, annot=True, cmap='coolwarm')
    plt.figure(figsize=(6,6))
    plt.title(pred_col + " Second-Order Sensitivities")
    sns.heatmap(second_Si, annot=False, square=True, cmap='coolwarm')
```

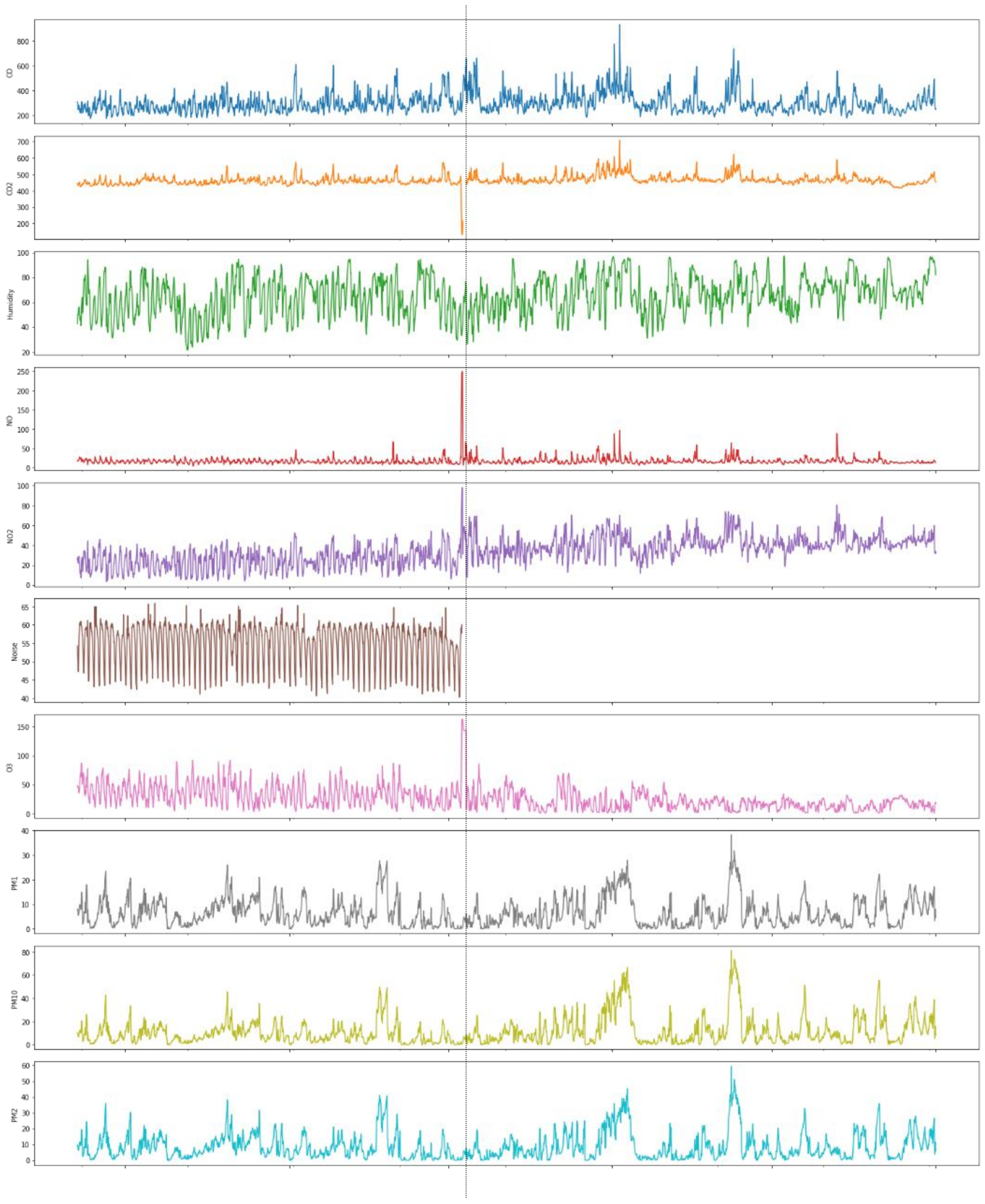
B. Data Visualization Plots

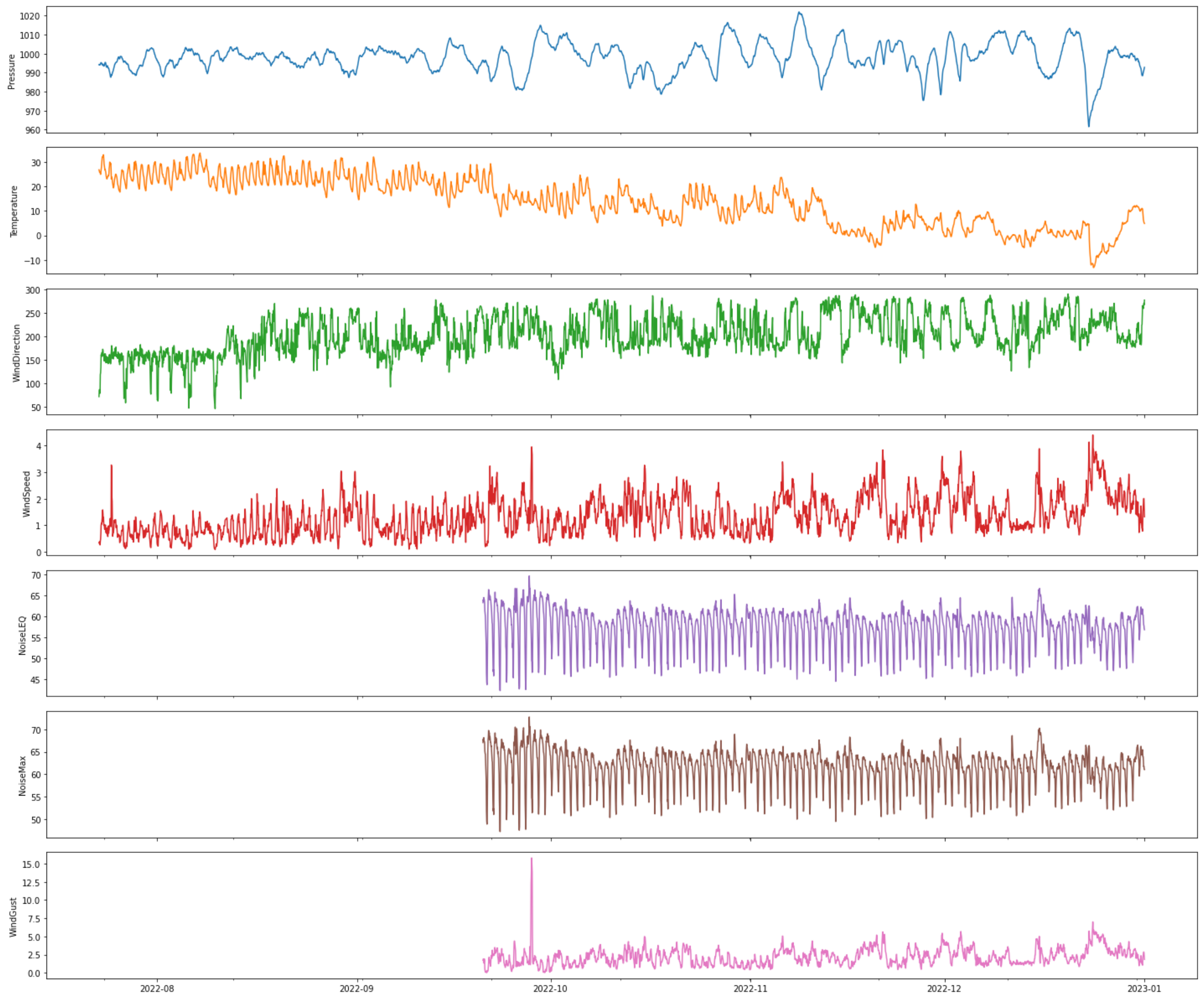
B.1 Visual Plot of 40-day, 1hr interval data




Elevation
Humidity
Latitude
NO
Pressure
Wind_Speed
Wind_Gust
Wind_Direction
Temperature
10
17
24
31
07
14
Nov
2022

B.2 Visual Plot of data from July 2022 – January 2023 (Oct 3rd split at dotted line)




Pressure
Temperature
WindDirection
WindSpeed
NoiseLEQ
NoiseMax
WindGust
2022-08
2022-09
2022-10
2022-11
2022-12
2023-01

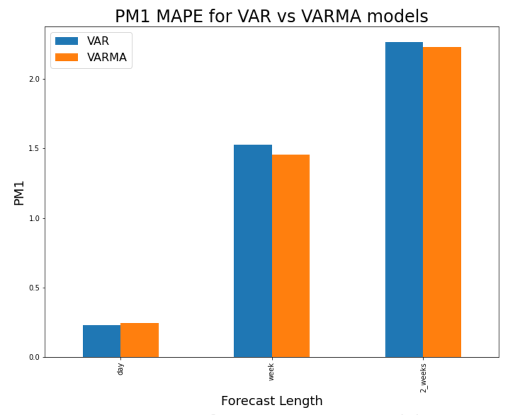
C. List of Trained Models

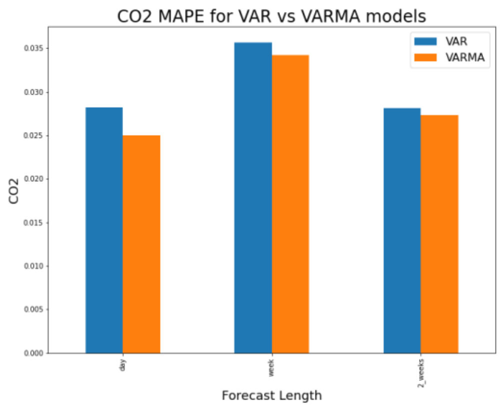
Index	Model Type	Device Location	Data Interval	Date Range	Training Size	Variables Used	Training Time
1	VAR	Y&E	60m	July-Oct	500	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed'], dtype='object')	92.42268
2	VAR	Y&E	60m	Oct-Jan	500	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed', 'NoiseLEQ', 'NoiseMax', 'WindGust'], dtype='object')	142.0159
3	VAR	Y&E	15m	July-Oct	500	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed'], dtype='object')	181.6118
4	VAR	Y&E	15m	Oct-Jan	500	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed', 'NoiseLEQ', 'NoiseMax', 'WindGust'], dtype='object')	144.7495
5	VAR	Y&E	1m	July-Oct	500	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindSpeed'], dtype='object')	140.9626
6	VAR	Y&E	1m	Oct-Jan	500	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed', 'NoiseLEQ', 'NoiseMax', 'WindGust'], dtype='object')	142.5576
7	VARMA	Y&E	60m	July-Oct	500	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection',	1715.332

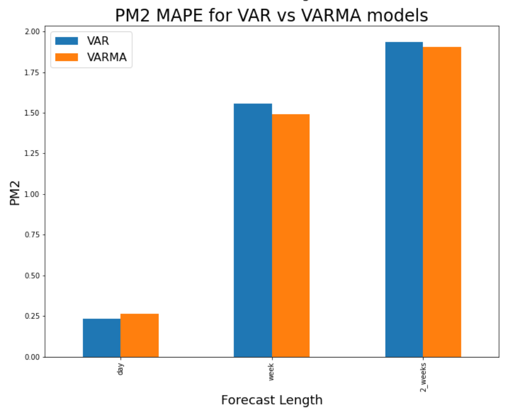
						'WindSpeed'], dtype='object')	
8	VARMA	Y&E	60m	July-Oct	500	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed'], dtype='object')	6343.357
9	VARMA	Y&E	15m	July-Oct	100	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed'], dtype='object')	1748.488
10	VARMA	Y&E	15m	July-Oct	300	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed'], dtype='object')	2593.323
11	VARMA	Y&E	15m	July-Oct	500	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed'], dtype='object')	274.401
12	VARMA	Y&E	15m	July-Oct	1000	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed'], dtype='object')	7853.08
13	VAR	Y&E	1m	July-Oct	300	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindSpeed'], dtype='object')	85.4404
14	VAR	Y&E	1m	July-Oct	500	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindSpeed'], dtype='object')	233.8613
15	VAR	Y&E	1m	July-Oct	1000	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindSpeed'], dtype='object')	3111.951

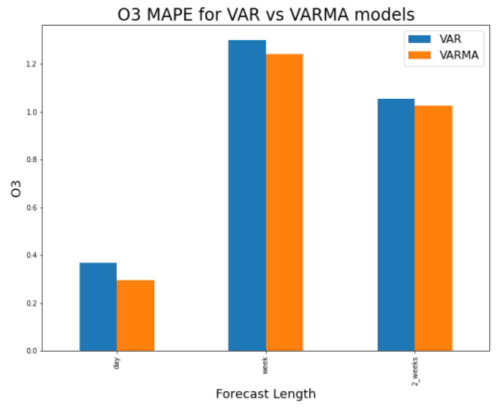
16	VAR	Y&E	1m	July-Oct	3000	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindSpeed'], dtype='object')	10510.14
17	VARMA	Y&E	60m	July-Oct	1000	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed'], dtype='object')	21387.68
18	VAR	Y&E	60m	July-Oct	1000	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed'], dtype='object')	458.6584
19	VAR	Y&E	60m	July-Oct	1000	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindDirection', 'WindSpeed'], dtype='object')	462.7447
20	VARMA	Y&E	1m	July-Oct	1000	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindSpeed'], dtype='object')	7376.368
21	VARMA	Y&E	1m	July-Oct	1000	Index(['CO', 'CO2', 'Humidity', 'NO', 'NO2', 'Noise', 'O3', 'PM1', 'PM10', 'PM2', 'Pressure', 'Temperature', 'WindSpeed'], dtype='object')	15988.87

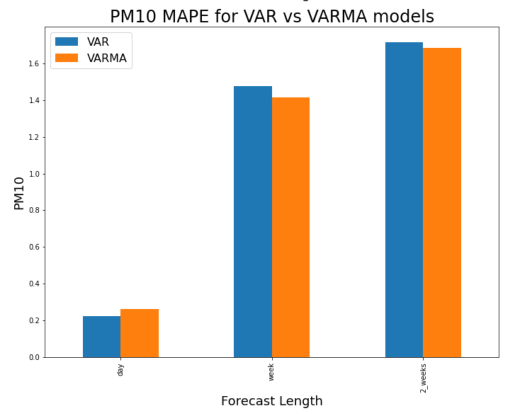
D. VAR versus VARMA MAPE for Individual Pollutants

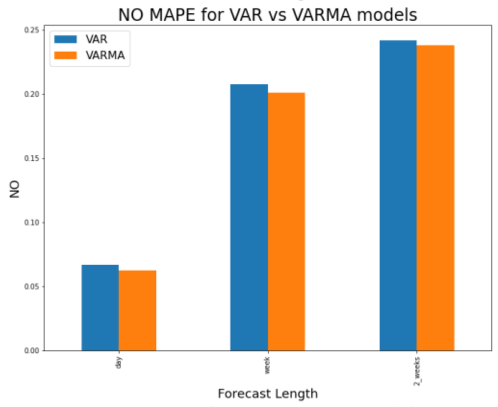


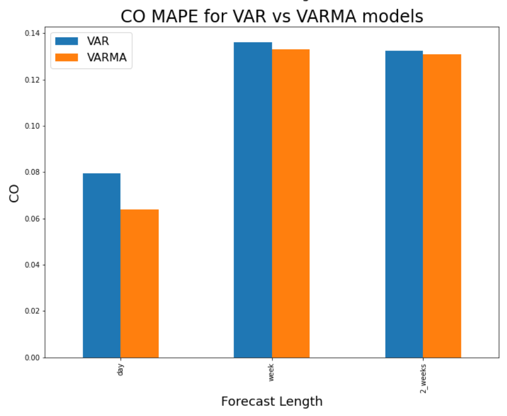


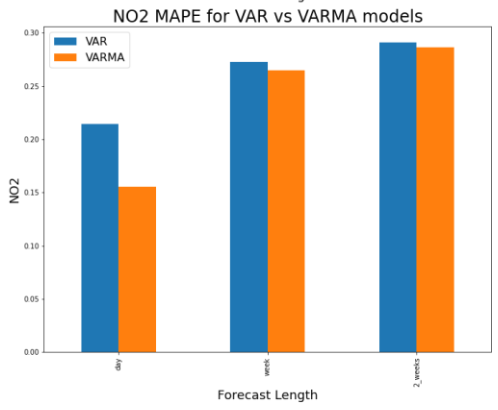




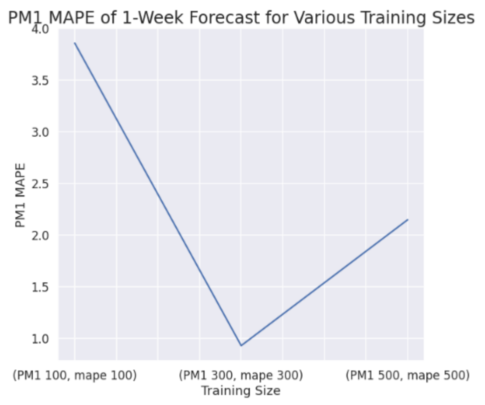


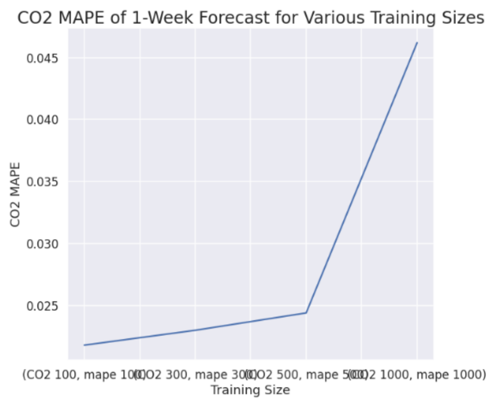


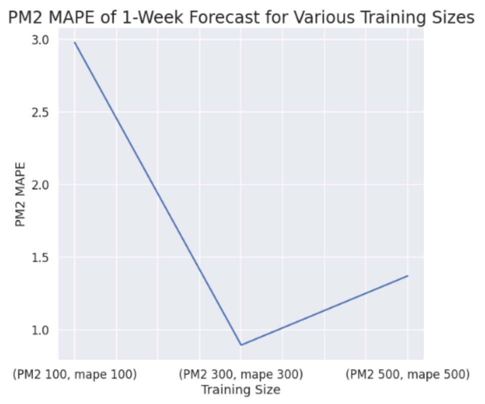


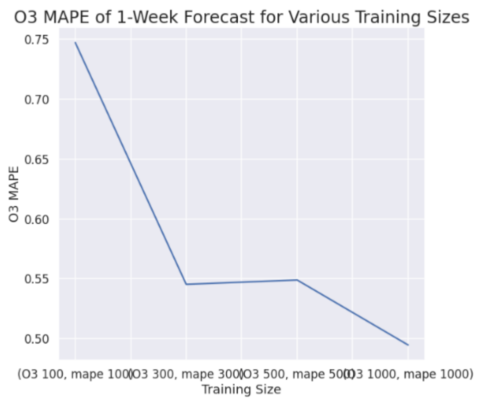


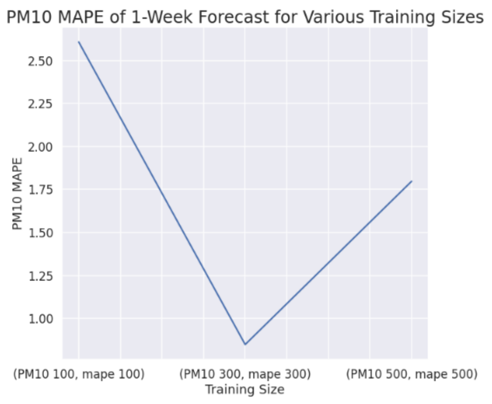
E. MAPE for Individual Pollutants for Various Training Sizes

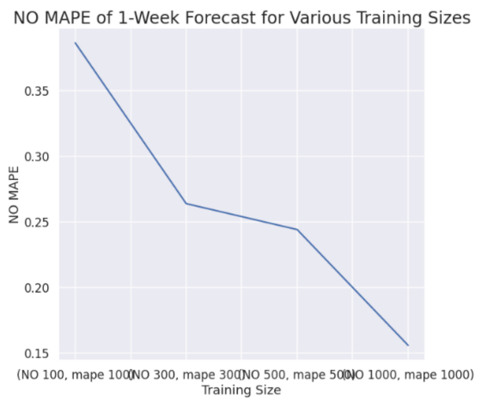


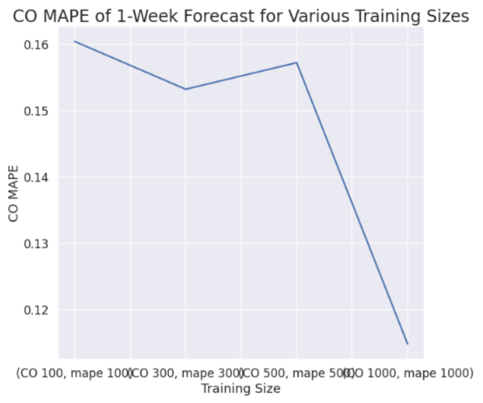


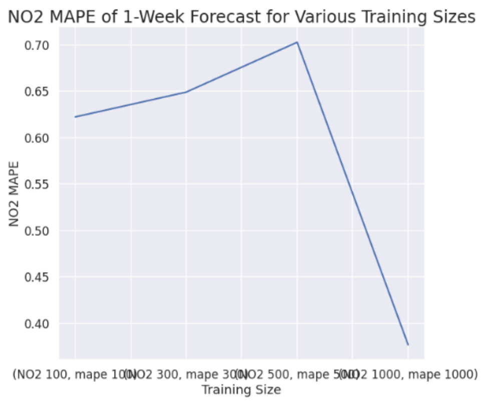






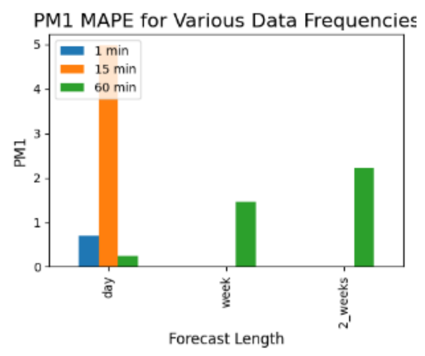


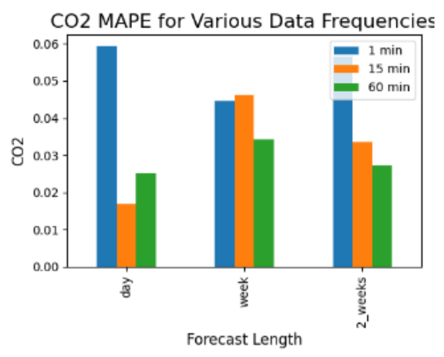


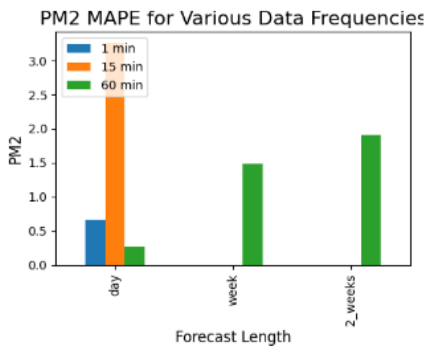


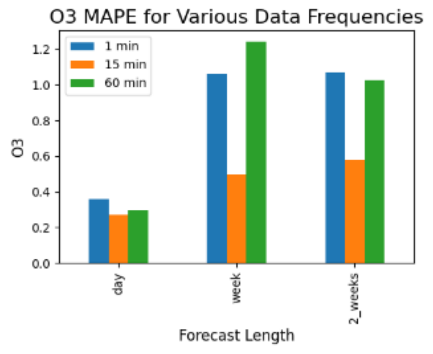
F. MAPE for Individual Pollutants for Various Data Resolutions

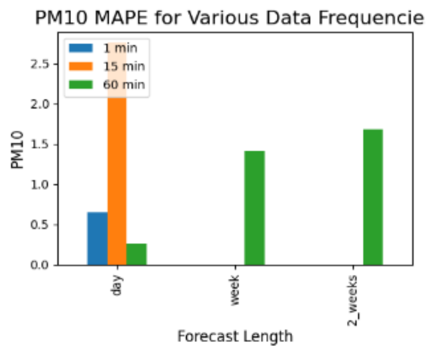
NOTE: Missing bars for Particulate Matter represent infinity MAPE values (due to close-to-zero observed values)

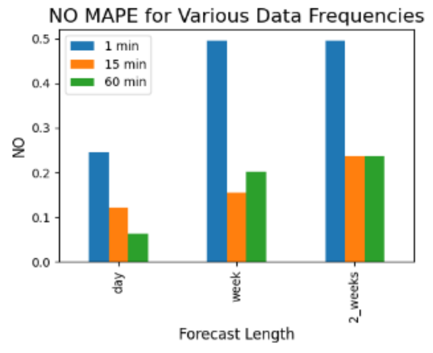


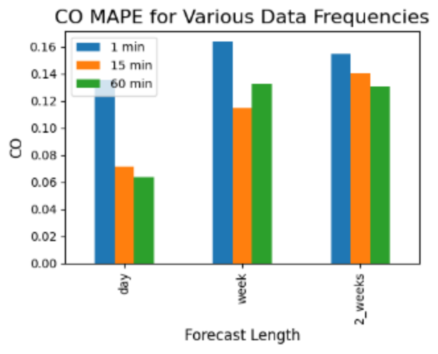


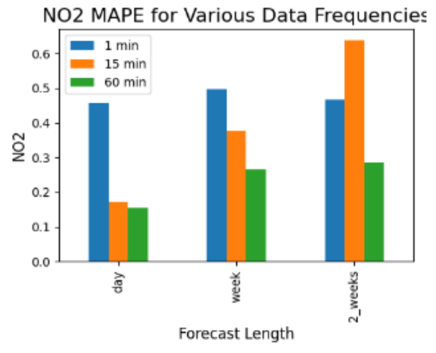




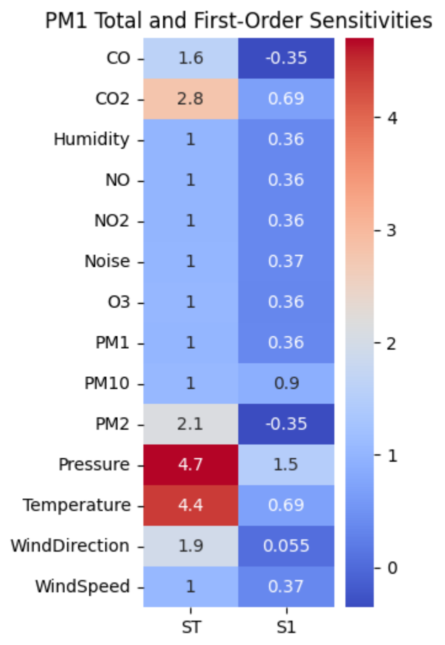


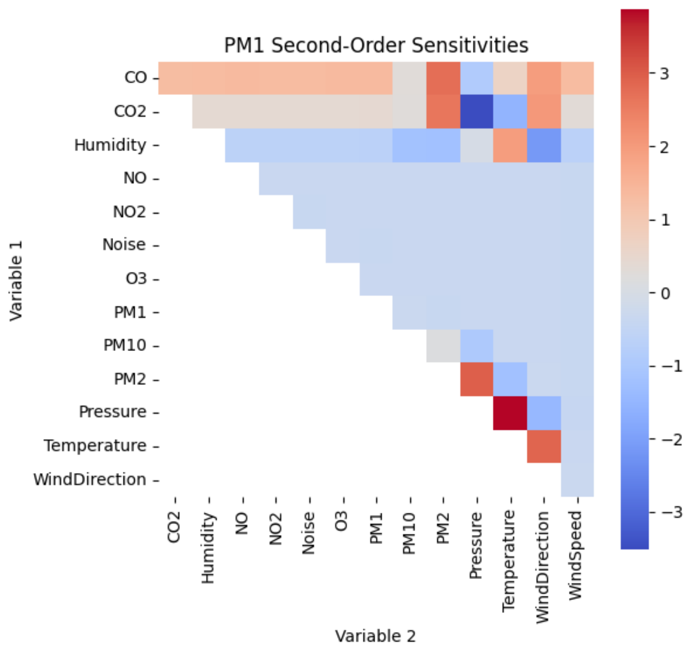


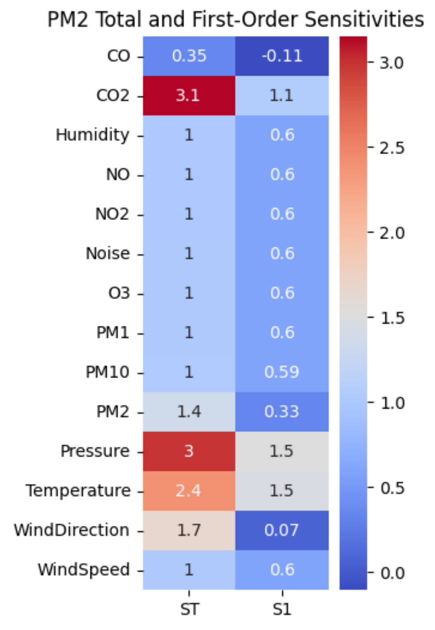


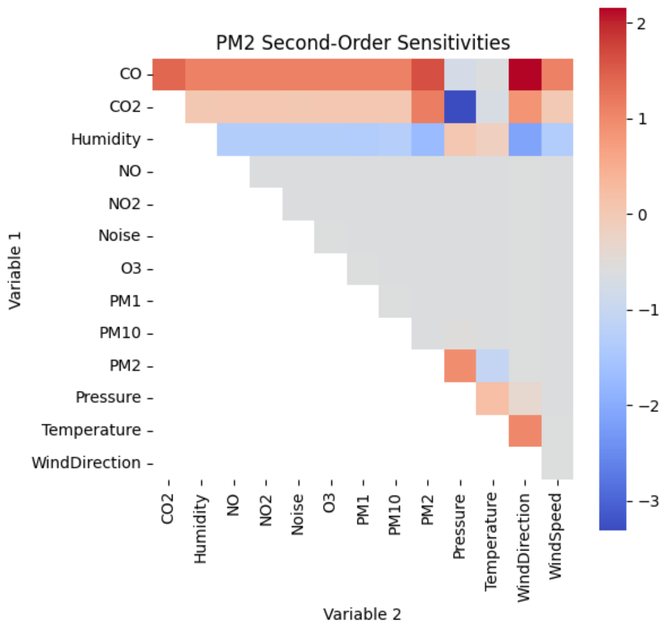


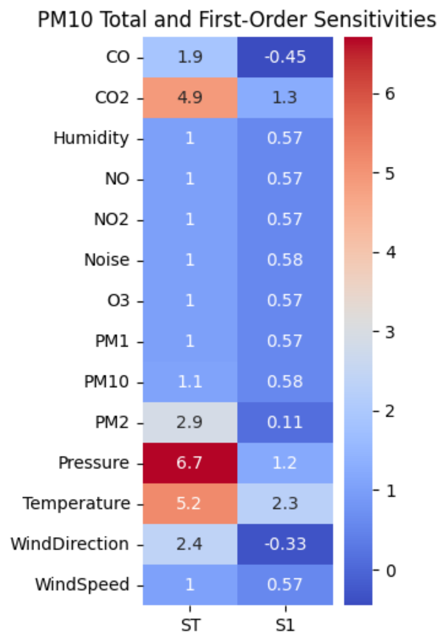
G. Sobol Sensitivity Analysis Results for Model #19 (VARMA, training_size=1000, resolution=60min)

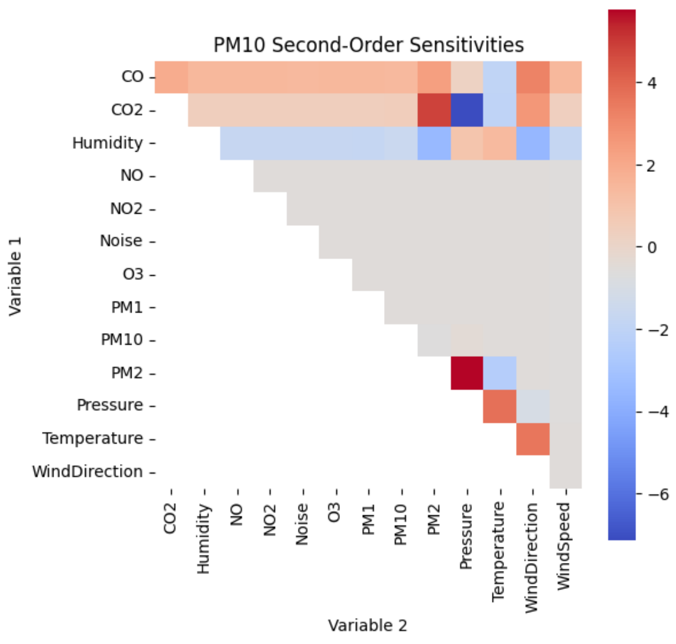


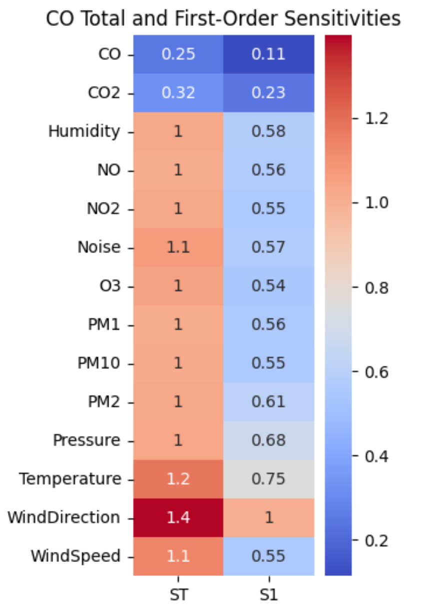


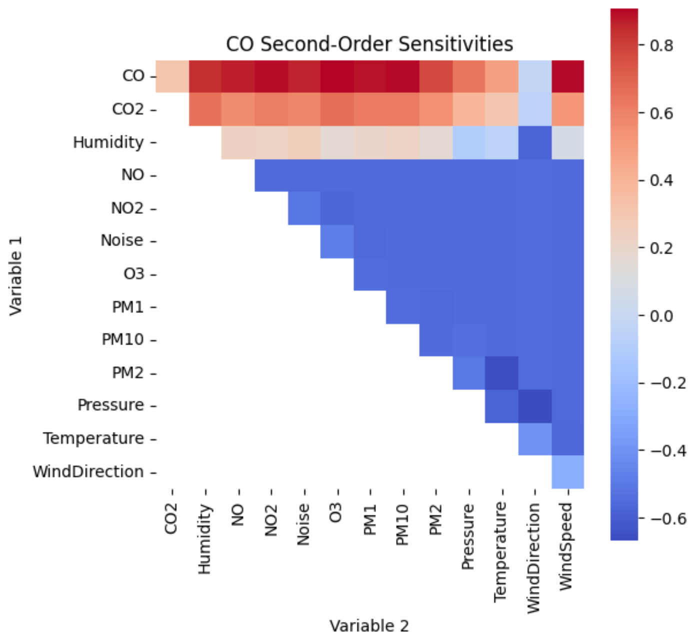


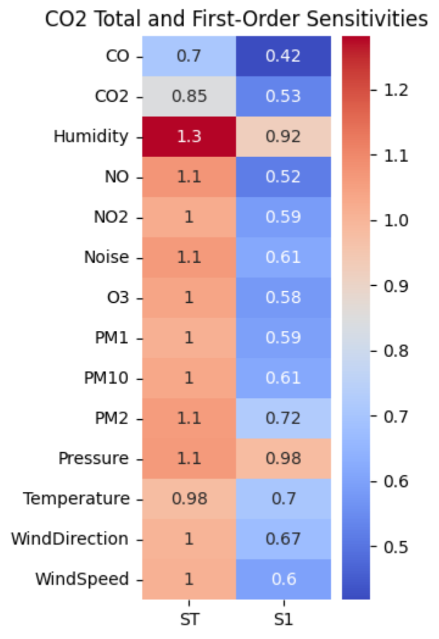


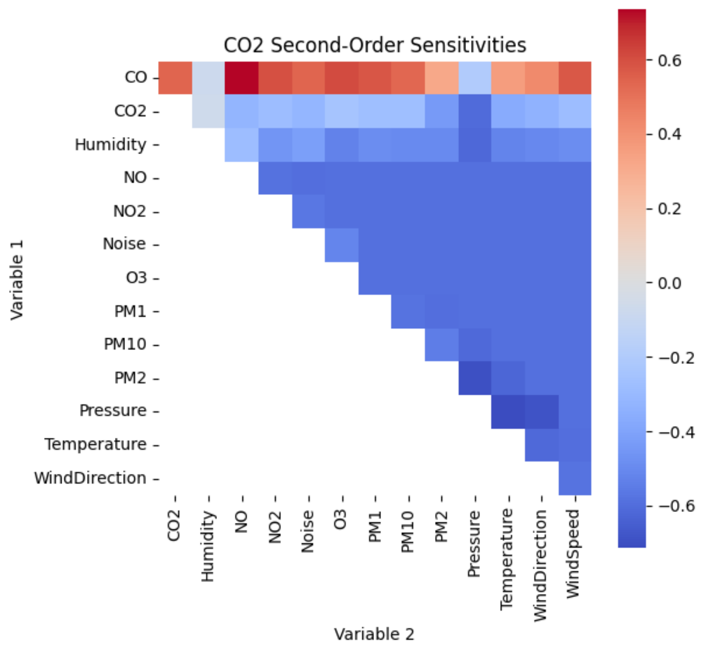

PM10 Total and First-Order Sensitivities
ST
S1
CO 1.9 -0.45
CO2 4.9 1.3
Humidity 1 0.57
NO 1 0.57
NO2 1 0.57
Noise 1 0.58
O3 1 0.57
PM1 1 0.57
PM10 1.1 0.58
PM2 2.9 0.11
Pressure 6.7 1.2
Temperature 5.2 2.3
WindDirection 2.4 -0.33
WindSpeed 1 0.57
6
5
4
3
2
1
0

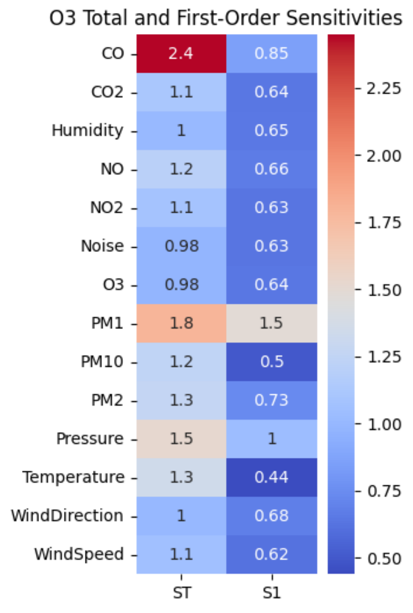

PM10 Second-Order Sensitivities
Variable 1
Variable 2
CO
CO2
Humidity
NO
NO2
Noise
O3
PM1
PM10
PM2
Pressure
Temperature
WindDirection
WindSpeed
4
2
0
−2
−4
−6

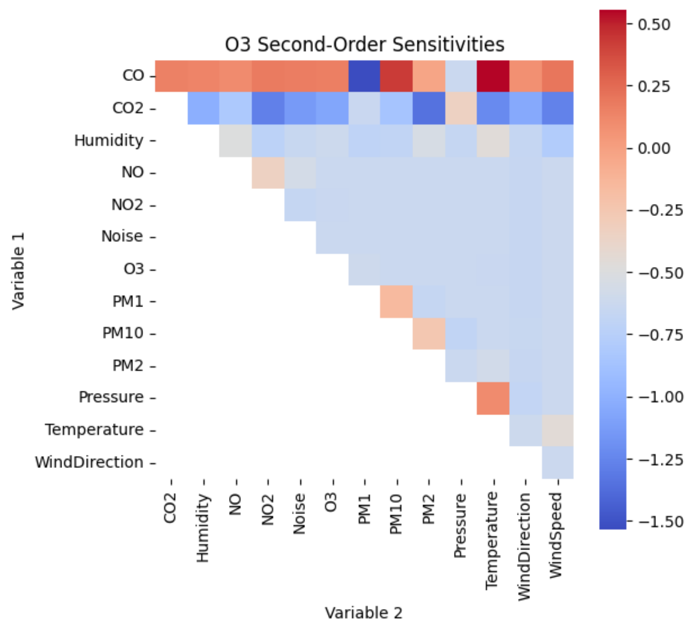

CO Total and First-Order Sensitivities
ST
S1
CO 0.25 0.11
CO2 0.32 0.23
Humidity 1 0.58
NO 1 0.56
NO2 1 0.55
Noise 1.1 0.57
O3 1 0.54
PM1 1 0.56
PM10 1 0.55
PM2 1 0.61
Pressure 1 0.68
Temperature 1.2 0.75
WindDirection 1.4 1
WindSpeed 1.1 0.55
1.2
1.0
0.8
0.6
0.4
0.2

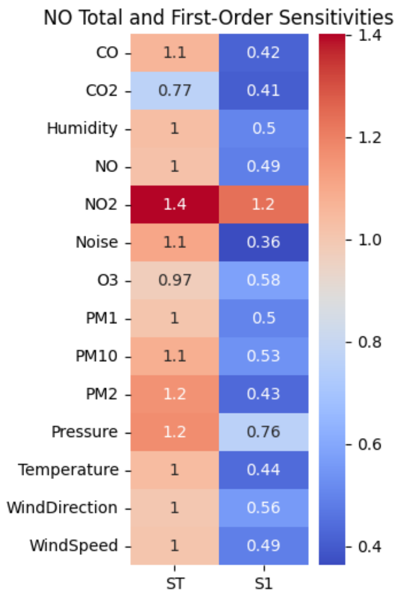

CO Second-Order Sensitivities
Variable 1
Variable 2
CO
CO2
Humidity
NO
NO2
Noise
O3
PM1
PM10
PM2
Pressure
Temperature
WindDirection
WindSpeed
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

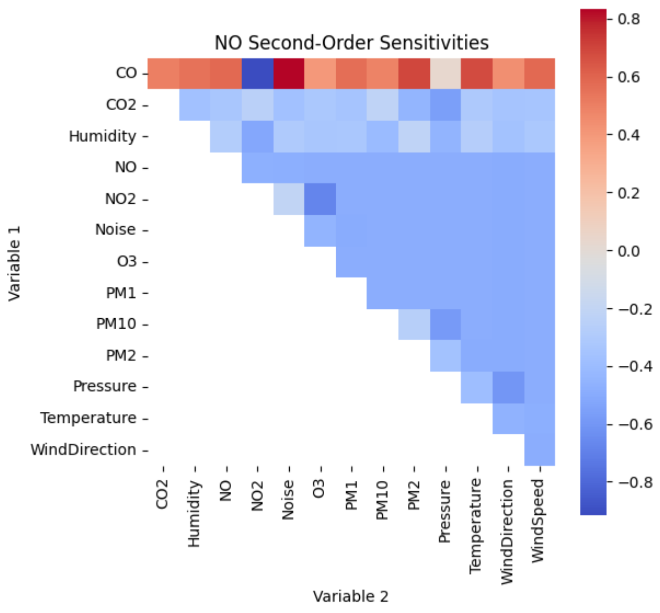

CO2 Total and First-Order Sensitivities
ST
S1
CO 0.7 0.42
CO2 0.85 0.53
Humidity 1.3 0.92
NO 1.1 0.52
NO2 1 0.59
Noise 1.1 0.61
O3 1 0.58
PM1 1 0.59
PM10 1 0.61
PM2 1.1 0.72
Pressure 1.1 0.98
Temperature 0.98 0.7
WindDirection 1 0.67
WindSpeed 1 0.6
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5

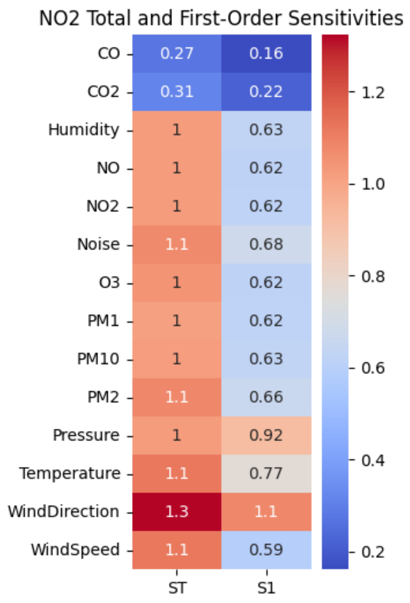

CO2 Second-Order Sensitivities
Variable 1
Variable 2
CO
CO2
Humidity
NO
NO2
Noise
O3
PM1
PM10
PM2
Pressure
Temperature
WindDirection
WindSpeed
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

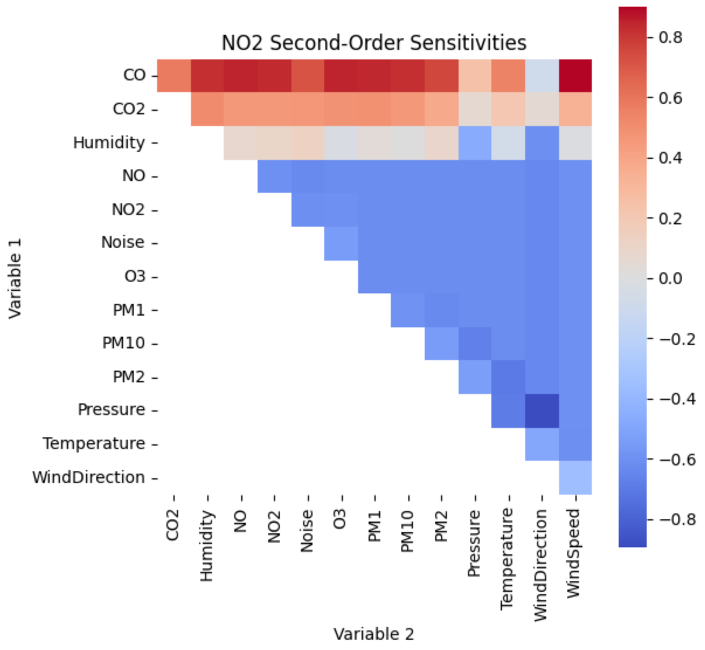

O3 Total and First-Order Sensitivities
ST
S1
CO 2.4 0.85
CO2 1.1 0.64
Humidity 1 0.65
NO 1.2 0.66
NO2 1.1 0.63
Noise 0.98 0.63
O3 0.98 0.64
PM1 1.8 1.5
PM10 1.2 0.5
PM2 1.3 0.73
Pressure 1.5 1
Temperature 1.3 0.44
WindDirection 1 0.68
WindSpeed 1.1 0.62
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50


O3 Second-Order Sensitivities
Variable 1
Variable 2
CO
CO2
Humidity
NO
NO2
Noise
O3
PM1
PM10
PM2
Pressure
Temperature
WindDirection
WindSpeed
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.50


NO Total and First-Order Sensitivities
CO 1.1 0.42
CO2 0.77 0.41
Humidity 1 0.5
NO 1 0.49
NO2 1.4 1.2
Noise 1.1 0.36
O3 0.97 0.58
PM1 1 0.5
PM10 1.1 0.53
PM2 1.2 0.43
Pressure 1.2 0.76
Temperature 1 0.44
WindDirection 1 0.56
WindSpeed 1 0.49
ST
S1
1.4
1.2
1.0
0.8
0.6
0.4

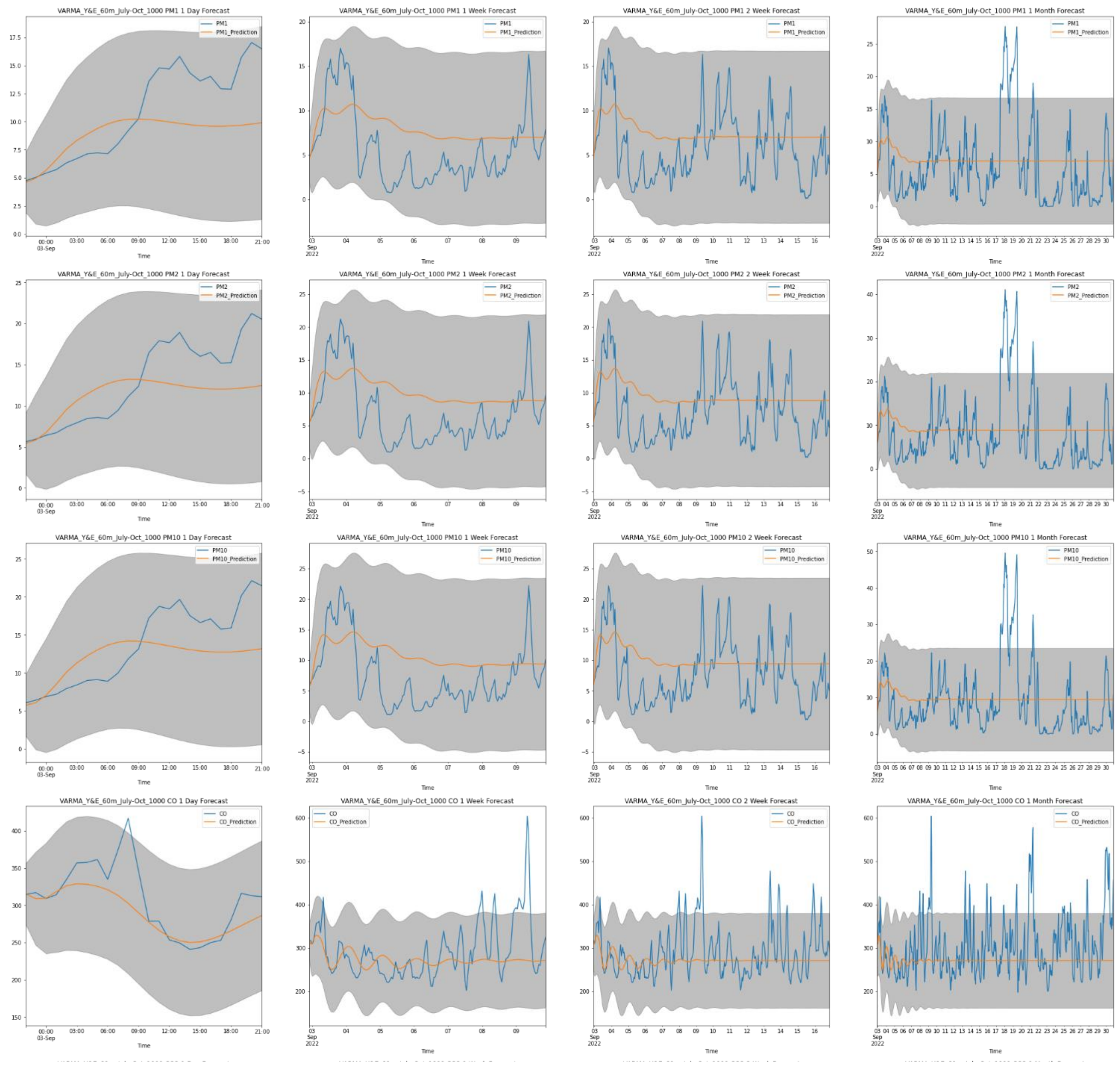

NO Second-Order Sensitivities
Variable 1
Variable 2
CO
CO2
Humidity
NO
NO2
Noise
O3
PM1
PM10
PM2
Pressure
Temperature
WindDirection
WindSpeed
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8

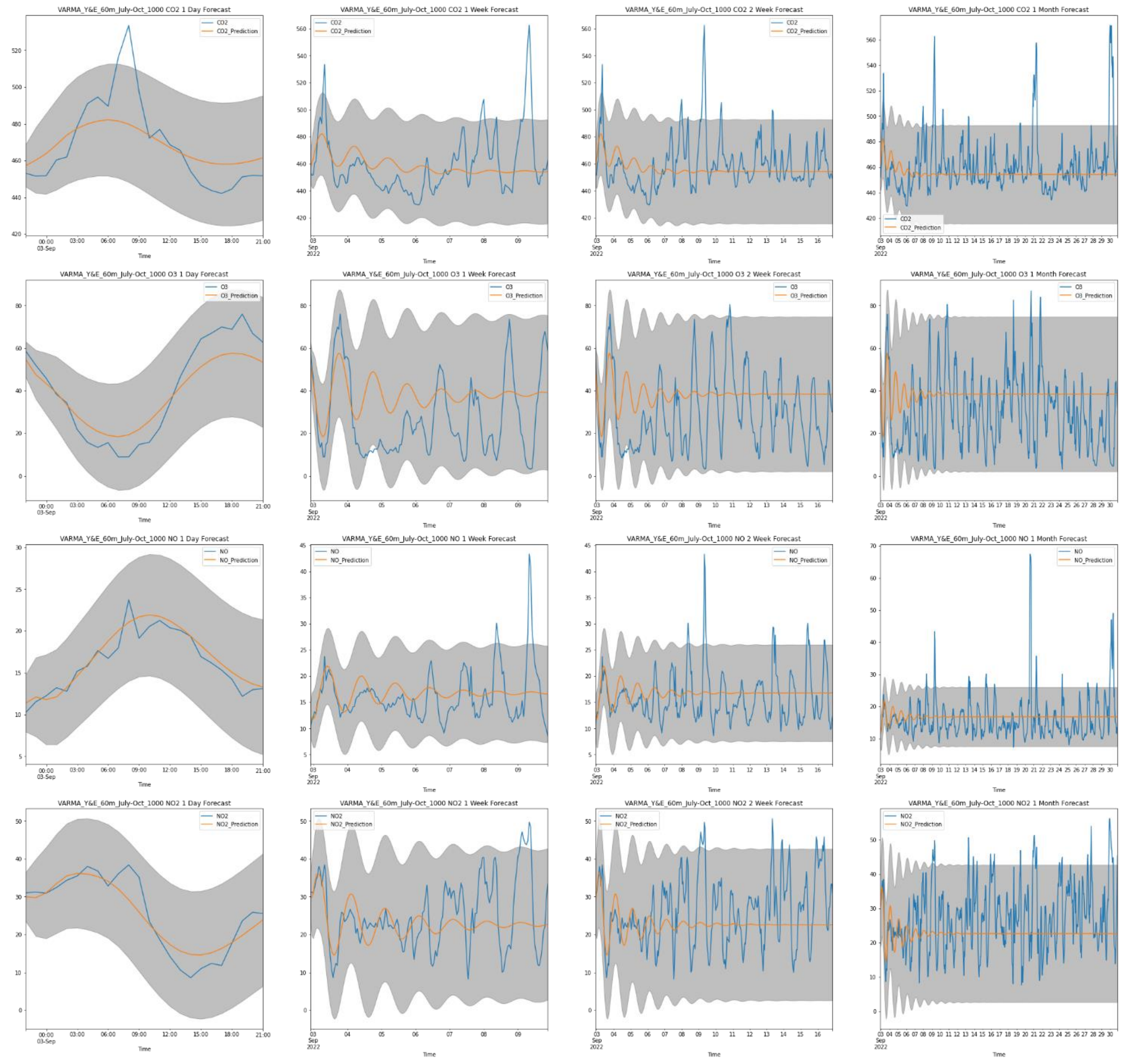

NO2 Total and First-Order Sensitivities
CO 0.27 0.16
CO2 0.31 0.22
Humidity 1 0.63
NO 1 0.62
NO2 1 0.62
Noise 1.1 0.68
O3 1 0.62
PM1 1 0.62
PM10 1 0.63
PM2 1.1 0.66
Pressure 1 0.92
Temperature 1.1 0.77
WindDirection 1.3 1.1
WindSpeed 1.1 0.59
ST
S1
1.2
1.0
0.8
0.6
0.4
0.2


NO2 Second-Order Sensitivities
Variable 1
Variable 2
CO
CO2
Humidity
NO
NO2
Noise
O3
PM1
PM10
PM2
Pressure
Temperature
WindDirection
WindSpeed
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8

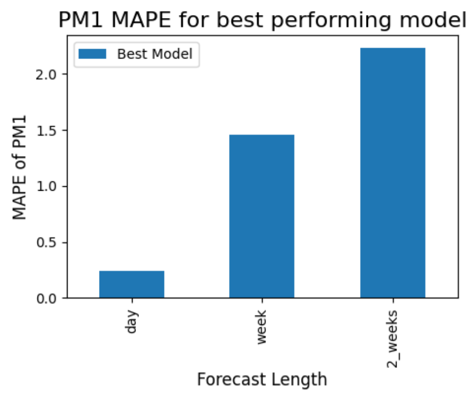
H. Best Performing Model Results (VARMA, training_size=1000, resolution=60m)

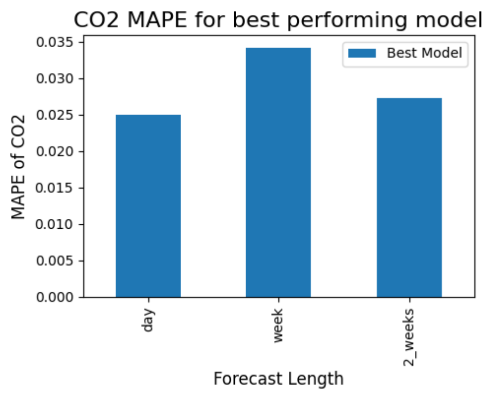
H.1 Plots Showing Prediction vs Actual Pollutant Levels for Varying Forecast Lengths

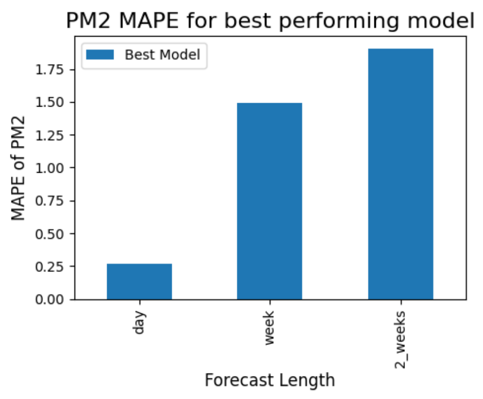


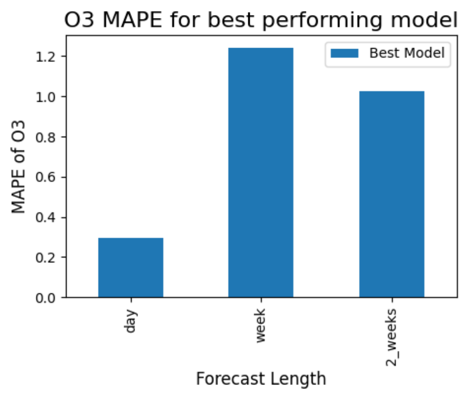

VARMA_Y&E_60m_July-Oct_1000 CO2 1 Day Forecast
VARMA_Y&E_60m_July-Oct_1000 CO2 1 Week Forecast
VARMA_Y&E_60m_July-Oct_1000 CO2 2 Week Forecast
VARMA_Y&E_60m_July-Oct_1000 CO2 1 Month Forecast
CO2
CO2_Prediction
VARMA_Y&E_60m_July-Oct_1000 O3 1 Day Forecast
VARMA_Y&E_60m_July-Oct_1000 O3 1 Week Forecast
VARMA_Y&E_60m_July-Oct_1000 O3 2 Week Forecast
VARMA_Y&E_60m_July-Oct_1000 O3 1 Month Forecast
O3
O3_Prediction
VARMA_Y&E_60m_July-Oct_1000 NO 1 Day Forecast
VARMA_Y&E_60m_July-Oct_1000 NO 1 Week Forecast
VARMA_Y&E_60m_July-Oct_1000 NO 2 Week Forecast
VARMA_Y&E_60m_July-Oct_1000 NO 1 Month Forecast
NO
NO_Prediction
VARMA_Y&E_60m_July-Oct_1000 NO2 1 Day Forecast
VARMA_Y&E_60m_July-Oct_1000 NO2 1 Week Forecast
VARMA_Y&E_60m_July-Oct_1000 NO2 2 Week Forecast
VARMA_Y&E_60m_July-Oct_1000 NO2 1 Month Forecast
NO2
NO2_Prediction
Time
Sep
2022
03-Sep

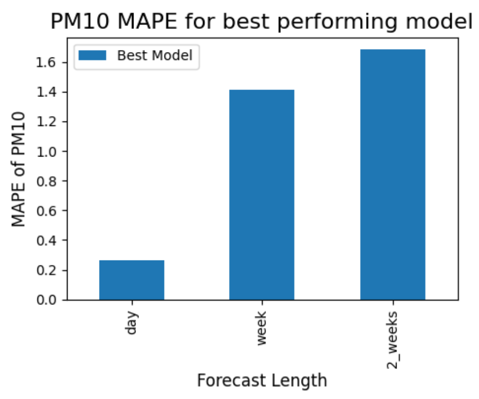
H.2 Mean Absolute Percentage Error (MAPE) for each individual pollutant

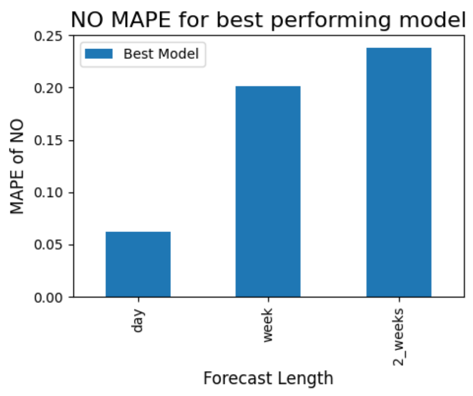

PM1 MAPE for best performing model
Best Model
MAPE of PM1
day
week
2_weeks
Forecast Length

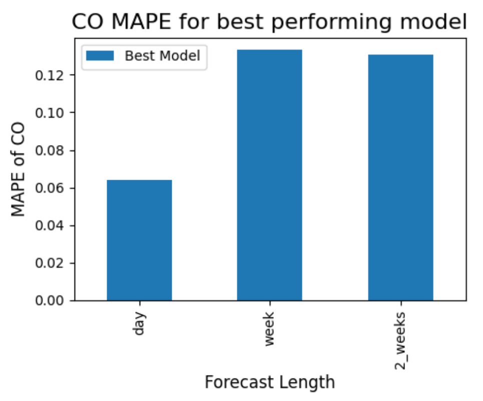

CO2 MAPE for best performing model
Best Model
MAPE of CO2
day
week
2_weeks
Forecast Length

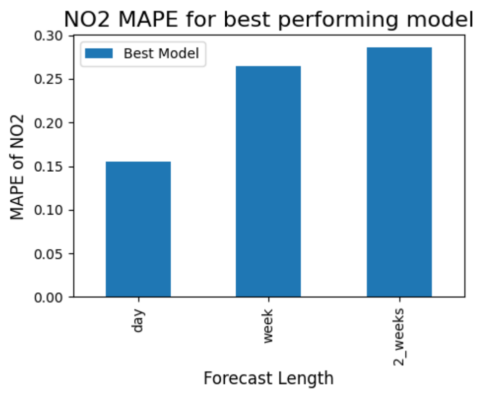

PM2 MAPE for best performing model
Best Model
MAPE of PM2
day
week
2_weeks
Forecast Length


O3 MAPE for best performing model
Best Model
MAPE of O3
day
week
2_weeks
Forecast Length


PM10 MAPE for best performing model
Best Model
MAPE of PM10
day
week
2_weeks
Forecast Length


NO MAPE for best performing model
Best Model
MAPE of NO
day
week
2_weeks
Forecast Length


CO MAPE for best performing model
Best Model
MAPE of CO
day
week
2_weeks
Forecast Length


NO2 MAPE for best performing model
Best Model
MAPE of NO2
day
week
2_weeks
Forecast Length