Forecasting Air Quality with the OpenAQ Atmospheric Pollution Dataset

Matthew J. Henry

Northwest Missouri State University, Maryville MO 64468, USA
s546783@nwmissouri.edu
matthewjhdec1991@gmail.com

Abstract. This work presents an approach for the use of public domain meteorological and air quality data to forecast air quality. Data was extracted from the OpenAQ and NWS(National Weather Service) daily summary data sets through the publicly available AWS (Amazon Web Services) S3 buckets to local storage. An approach for cleaning and storing the data in a locally hosted data warehouse along with pairing locations from the two separate datasets is presented. Proof of concept predictive time-series and LSTM(Long Short-Term Memory) neural net models are then developed on a subset of the data using meteorological variables. Forecast models with reasonably accurate predictions over a 24 hour window are presented, and limitations of the modeling approach are discussed. Autoregressive models using meteorological exogenous variables demonstrated superior performance to the LSTM and other regression models evaluated.

Keywords: AWS, Time Series, PM2.5, Ozone, Nitrogen Dioxide, LSTM, Weather, OpenAQ, Autoregressive Models

1 Introduction

The main domain of analysis is to explore, through the lens of data analytic techniques, the predictive utility of weather data on the concentrations of atmospheric pollutants. Air pollution has been tied to significant adverse health events and morbidity [9] as well as mental health impacts [13]. It is also established that a fair degree of spatial resolution is necessary for predicting health consequences [14]. Furthermore there are examples in the literature such as Cordova et. al. that demonstrate successful application of neural networks to forecasting air quality [8], providing motivation for the utility of data science techniques in this domain.

1.1 Data Sources of Interest

OpenAQ maintains a repository of air quality data from over 158 different countries, aggregating data over PM2.5 (Particulate Matter two and a half

microns or less in width), PM10 (Particulate Matter less than 10 micrometers in width), ozone, sulfur dioxide, nitrogen dioxide, carbon monoxide, and black carbon. Research and government measurements as well as low cost sensor data are gathered. The project also maintains an API for access to recently gathered data (`https://openaq.org/`)[1]. The national weather service also maintains a daily summary of weather for all weather stations. This data can be accessed from the AWS (Amazon Web Services) Registry of Open Data (`https://registry.opendata.aws/noaa-gsod/`)[5].

1.2 Data Analysis Objectives

The intended goal is to evaluate the impact that weather conditions have on the concentrations of various atmospheric pollutants, and explore the utility of such variables on the forecasting of air quality. Regression analysis, time series forecasting techniques, and neural network methodology will be explored as potential modeling approaches. The main aim will be to develop simple easily trained models that only require data for one location and can be prepared using only the data available in the public domain from the data sources discussed above. Initially it was considered to add additional variables describing human influence on the environment, and to explore impact on public health metrics; however, considering time constraints and the difficulty of the undertaking it was decided to to limit this project to exploring the interplay between weather and air quality and to focus on forecasting methodology.

1.3 Outline of Project Implementation

The project commenced with the extraction of the air quality data from OpenAQ through the use of Athena (an Amazon Web Series query platform). The weather data was downloaded to local storage using the AWS(Amazon Web Services) CLI(Command Line Interface).

Afterwards a data validation and cleaning process was conducted. The data was aggregated, cleaned, and transformed to make it amenable to efficient storage in a database. After cleaning the data was imported into a PostgreSQL database. All data manipulations were performed and recorded in Jupyter notebooks to ensure documentation and reproducibility.

Initial data exploration was carried out using Tableau on selected data extracted using SQL queries. Variables were evaluated for correlation with air quality measurements. Descriptive statistics describing these correlations were computed.

Models were then constructed using variables identified as signficant during exploratory data analysis. The models were limited by the scope and resolution of publicly available data, which was not routinely available more frequently than on a daily basis.

1.4 Hardware and Software Used

The python version used was 3.9.13. Keras version 2.9.0 was used, and the sklearn version was 1.0.2. The Anaconda python distribution was used, and all packages were obtained using conda when available (all other packages were obtained using pip). The computer operating system was Windows 11. Exploratory data analysis was carried out using Tableau 2022.3. PostgreSQL version 15.1 was used as the database server.

All data cleaning and model training was carried out on two computers. One computer had a R9 3900X (12 core, 24 thread) with 64 GB of memory and the other computer an R9 7950X (16 core, 32 thread) and 64 GB of RAM. No GPU acceleration was utilized when using tensorflow/keras. Users looking to rerun the provided notebooks will be able to run them with any modern CPU, but 64 GB of RAM will be required (some of the cleaning code allocates on the order of 60 GB of RAM).

2 Data Extraction

Climate data was extracted from the OpenAQ AWS S3 buckets (`https://registry.opendata.aws/openaq/`) [2]. Querying with Athena it was found that no entries for the united states existed before 2014. After examining the size of the data set, it was deemed appropriate to extract the entire dataset (US only) from 2014-October 2022 for local manipulation, cleaning, and database construction.

To extract the data an S3 bucket was created with an external table to receive the results. I used the setup provided by Jonathan Padilla on Medium as a template to begin[12].

The data was in a primarily tabular format with some fields containing JSON like strings. The verification that these were not valid JSON and the process through which the fields were remediated will be discussed in the methodology section below. It was, however, necessary to adjust the ignore malformed json tag value to true in the ddl file in order to prevent the query from failing, which will be available on my github repository (`https://github.com/matthew-henry/Data-Analytics-Capstone-Project/tree/main/Extracted%20Air%20Quality%20Data`).

It was necessary to experiment with query parameters for data extraction via Athena to avoid timeouts. A bulk extraction of all US data was not feasible. It was, however, possible to extract the data in 1 year increments. Given that the threshold for time intervals under which the queries would timeout was not known, the queries were issued manually using the AWS cloudshell. A sample template query and screenshots of the process are included below:

```
SELECT * FROM default.openaq_jsonfix
WHERE country='US' AND
(date.utc BETWEEN '20XX-12-31' AND '20YY-01-01');
```

Fig. 1. Creation of the ddl file for the Athena queries in the AWS cloudshell environment.

Fig. 2. Issuing Athena queries in the AWS cloudshell environment.

Fig. 3. Checking for the completion of the last issued query in the cloudshell environment.

After the queries were completed I downloaded the results (stored as a CSV file) from the S3 bucket to local storage for processing and database construction. These files were downloaded from the AWS console through the web, sample screenshots of the process are provided below.

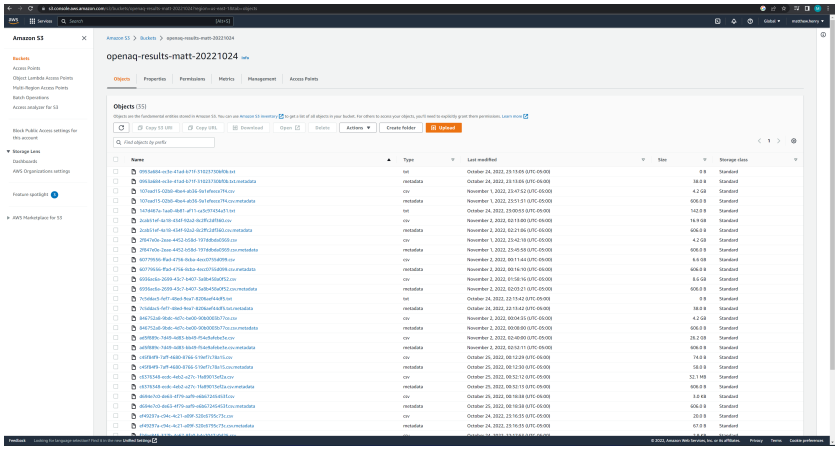

Fig. 4. Viewing files in the S3 bucket for the project.

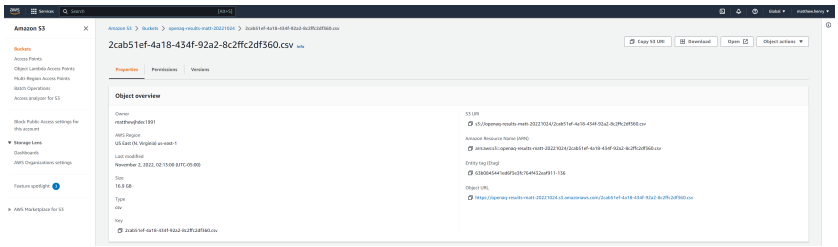

Fig. 5. Downloading a file from the S3 bucket.

It was also necessary to find a weather data set that could be easily coupled with the climate data had been gathered. AWS samples provided a project linking an open NOAA(National Oceanic and Atmospheric Administration) dataset hosted on AWS to OpenAQ data [7]. The NOAA Global Surface Summary of Day data set is available from AWS S3 buckets from the Registry of Open Data (`https://registry.opendata.aws/noaa-gsod/`)[5]. The data was downloaded using the AWS CLI(command line interface) tools installed locally. The data was organized into different folders by year of collection(from 2016 to 2022. A template of the s3 command used for extraction and a screenshot of the process are provided below. A batch file automating the process has been made available for Windows systems on this project's github repository(`https://github.com/matthew-henry/Data-Analytics-Capstone-Project/blob/main/Extracted%20Air%20Quality%20Data/AWS%20Weather%20Data%20Extraction.bat`.

```
aws s3 sync "s3://noaa-gsod-pds/20XX" .
```

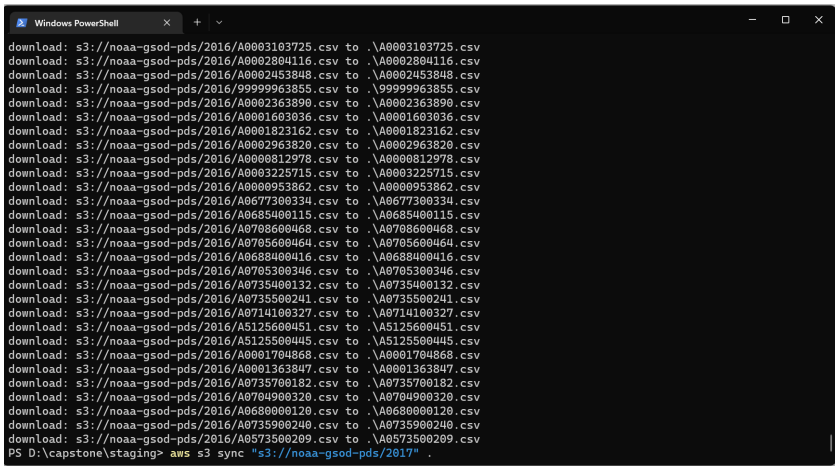

Fig. 6. Download of S3 bucket contents for Global Surface Summary of the Day data.

The weather data from the NWS is provided as a collection of CSV files reporting from various reporting stations. The data set includes the mean, maximum, and minimum for daily temperatures. It also contains the mean sea level pressure, mean station pressure, mean visibility, mean wind speed, maximum sustained wind speed, maximum wind gust, precipitation amount, and snow depth. The data set is documented by the NCEI (National Centers for Environmental Information) and also includes explanation of how non-reported data points are typically encoded[4].

3 Methodology (Cleaning and Data Extraction)

The air quality data was extracted in a set of CSV files ranging from 500 KB to 27.5 GB in size, and the weather station data was split between approximately 85000 CSV files of around 100 KB in size. The overall strategy was to clean the air quality data files individually (due to their large sizes) while the weather data were merged by year and then cleaned.

All data cleaning was performed in Jupyter notebooks using Python and the Pandas library. All code and output (but not data) can be found on this project's github repository (`https://github.com/matthew-henry/Data-Analytics-Capstone-Project`).

3.1 Cleaning the Air Quality Data

The columns in the tables extracted from AWS Athena contained a date column, parameter column, location column, value, column, unit column, city column, attribution column, averaging period column, coordinates column, country column, sourcename column, sourcetype column, and mobile column. The

date, attribution, averaging period, and coordinates column contained JSON-like strings. The list below describes the contents of the columns. The cleanup notebooks for the air quality data are found in the "Extracted Air Quality Data" folder in the github repository (`https://github.com/matthew-henry/Data-Analytics-Capstone-Project/tree/main/Extracted%20Air%20Quality%20Data`).

- date - Field contains the utc and local datetime stamp.
- parameter - Field contains the air contaminant species being measured.
- location - Field contains a description of the location where the measurement was made.
- value - Field contains the measured quantity of the pollutant measured.
- unit - Field contains the units of measurement (ppm and etc).
- city - Field contains the city where the measurement was made.
- attribution - Field contains the name and url of the group making the measurement.
- coordinates - Field contains the latitude and longitude of the measurement location.
- country - Field contains the country where the measurement was made.
- sourcename - Field contains the name of the source group. This field is not necessarily identical to the data in the attribution field.
- mobile - A boolean field describing whether the data was gathered from a boolean source.

Addressing the JSON-like Fields The first order of business was to inspect the key-value encoded text fields. The head of the tables had fields of the form key1=value1, key2=value2 which while similar is not valid formatting for a JSON-string. I wrote a function to verify whether the JSON string could be deserialized, but the process failed on every entry. As such the first cleaning step was to turn these strings into valid JSON.

The conversion process was relatively straight forward as the text was well structured. The first step would be to remove any starting or trailing bracket characters. Then the string would be split into separate strings at the commas. At each substring the "=" character would be replaced with a colon and the key and value would be wrapped in quotation marks. The string would then be recombined with a comma delimiter and the opening "" and closing "" added back on. The function for this transformation is included below.

```
def correct_field_to_json(arg: str) -> str:
    string = arg.lstrip("{{")
    string = string.rstrip("]}")
    strings = string.split(",")
    new_strings = []
    for entry in strings:
        index = entry.find('=')
        entry = '"' + entry[0:index] + '"' + ':' + '"'+entry[index+1:len(entry)]+'"'
```

```
        new_strings.append(entry)
    output_string = (',').join(entry for entry in new_strings)
    output_string = '{' + output_string + '}'
    return output_string
```

Upon rerunning the JSON validation loop the entries were verified to have all been converted into a deserializable format. Despite some initial concerns about the potential performance of such a function, the code executed in a very manageable time frame even on source files in excess of 10 GB in size.

The next step in processing this data was to reduce these JSON entries to single entry columns. The first naive approach attempted was to apply the JSON deserializer to each element, invoke the pandas Series constructer to flatten the fields, and then concatenate them together. Such a transformation is demonstrated in the code block below. However, this process was extremely slow and memory inefficient (even 4 GB input files would lead to python consuming in excess of 50 GB of RAM).

```
df = pd.concat([df, df.column.apply(json.loads).apply(pd.Series)], axis = 1)
```

In order to process my data within hardware limits for memory and in a reasonable time period, it was necessary to find an optimization for this process. Switching to a more performant json deserialization library (ujson) had negligible effect on performance. Similar performance issues in tasks had been documented, benchmarked, and profiled by Schönleber and indicated the use of expensive functions being passed to df.apply()[15]. Substantial memory usage reductions were achieved by leveraging the json normalize method in the pandas library. The final code included below was on the order of more than twenty times faster on large data sets.

```
expanded_datascope = df['column'].apply(ujson.loads)
expanded_datascope = pd.json_normalize(expanded_datascope)
df = pd.concat([df, expanded_datascope], axis =1)
```

Cleaning Outcomes and Preparation for Database import. After all the data had been processed and split into separate columns all redundant or unnecessary columns were dropped. The columns were renamed where necessary and the data was exported to a CSV file. The import process and final structure of the tables will be discussed in a later section. Flattening the JSON fields and removing some unnecessary text data caused a substantial reduction in file size as well as leaving it in an easily query-able format. The file sizes of the inputs and outputs are tabulated below.

3.2 Cleaning the Weather Data

The S3 bucket containing the weather data from NOAA was organized only in folders where all CSV files for an entire year were stored named by the observation stations numerical id. This lead to the extracted data being split between

Year	Original	Cleaned
2014	296 KB	220 KB
2015	31,424 KB	18,218 KB
2016	4,355,426 KB	2,223,199 KB
2017	4,362,612 KB	2,252,040 KB
2018	4,384,291 KB	2,265,808 KB
2019	6,965,347 KB	3,596,281 KB
2020	9,010,152 KB	4,648,094 KB
2021	17,694,811 KB	9,133,782 KB
2022	27,502,785 KB	13,855,190 KB

tens of thousands of files. The first step in processing the data was to merge the files in each year. This merger was accomplished using Python in a Jupyter notebook, and the code can be found in the github repository folders for weather data cleaning (`https://github.com/matthew-henry/Data-Analytics-Capstone-Project/tree/main/Weather%20Data`). The columns in the CSV's are detailed as follows.

- STATION - The numerical station ID of the reporting station.
- DATE - The date that the observations were made.
- LATITUDE - The latitude of the observation station.
- LONGITUDE - The longitude of the observation station.
- ELEVATION - The elevation in meters.
- NAME - Name of reporting station
- TEMP - Mean temperature for the day in degrees Fahrenheit.
- TEMP ATTRIBUTES - Number of observations averaged together.
- DEWP - Mean dewpoint for the day in degrees Fahrenheit.
- DEWP ATTRIBUTES - Number of observations averaged together.
- SLP - Mean sea level pressure in millibars.
- SLP ATTRIBUTES - Number of observations averaged together.
- STP - Mean station pressure for the day in millibars.
- STP ATTRIBUTES - Number of observations averaged together.
- VISIB - Mean visibility for the day in miles.
- VISIB ATTRIBUTES - Number of observations averaged together.
- WDSP - Mean wind speed for the day in knots.
- WDSP ATTRIBUTES - Number of observations averaged together.
- MXSPD - Maximum sustained wind speed reported for the day in knots.
- GUST - Maximum wind gust reported for the day in knots.
- MAX - Maximum temperature reported for the day in Fahrenheit.
- MAX ATTRIBUTES - Denotes whether it was an explicit maximum or derived from hourly data.
- MIN - Minimum temperature reported for teh day in Fahrenheit.
- MIN ATTRIBUTES - Denotes whether it was an explicit maximum or derived from hourly data.
- PRCP - Total precipitation for the day in inches of rain or melted snow or ice.
- PRCP ATTRIBUTES - Contains specifics about collection methodology.

- SNDP - Depth of snow in inches.
- FRSHTT - A field containing whether fog, rain, snow, hail, thunder, or a tornado occured for the day.

Once merged data cleaning was straightforward and involved handling missing data and dropping unneeded columns. Missing data points that were usually zero were set to zero, and all other missing data columns were set to null (the source data encoded missing data points with specific numerical values). The only significant transformation required was to convert the categorical FRSHTT column. The original column contained data encoded in a string sequence of 0's and 1's, this was converted to text entries delimited by a "—". The details of the files encoding are detailed in the NCEI's readme file[**?**]. After the data was aggregated the attributes columns were removed and the data was exported as a CSV file.

3.3 Construction and Structure of the Database

Two tables were created in postgres one for the air quality data (aqtable) and one for the weather data (wxtable). The names and the data types of the two tables are below. The CSV files were imported into the SQL database after the tables had been created.

Air Quality Table Data Structure

- parameter - text - name of pollutant measured.
- location - text - contains a description of the location of the measurement.
- value - float - contains the measured concentration.
- concentration units - text - contains the units of measurement.
- city - text - contains the city where teh measurement was made.
- country - text - contains the country where the measurement was made.
- sourcename - text - contains the name of the measuring party.
- sourcetype - text - what type of entity made the measurement.
- latitude - float - latitude
- longitude - float - longitude
- utc - timestamp - the time of the measurement in UTC
- local - timestamp - the time of the measurement in the local time zone
- source - text - source of the measurement
- url - text - URL of the measuring party
- averaging time unit - text - the units of the averaging time interval
- averaging time - text - how long the concentration data was averaged for

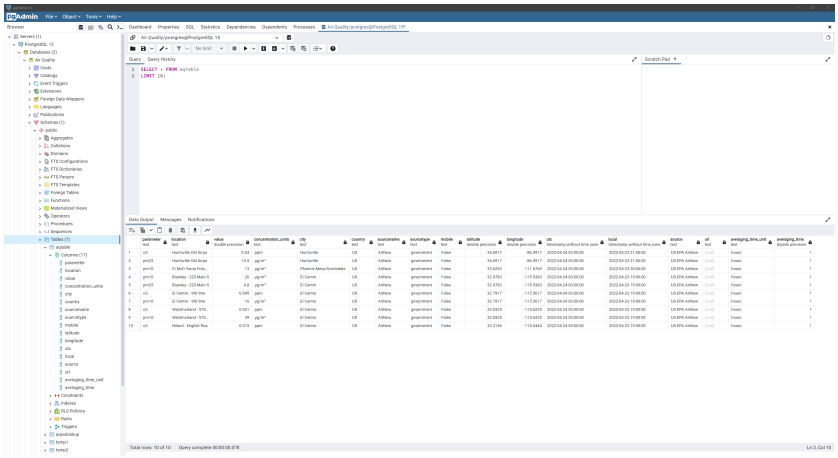

Fig. 7. First 10 rows of aqtable from database.

Weather Table Data Structure

- station - text - the station number for the measurement
- date - date - the date of the measurement
- latitude - float - latitude
- longitude - float - longitude
- elevation - float - the elevation of the measuring station in meters.
- name - text - name of reporting station
- temperature - float - the mean temperature for the day
- dewpoint - float - the mean dewpoint for the day
- slp - float - the mean sea level pressure in millibar
- stp - float - the mean station pressure in millibar
- visibility - float - the mean visibility for the day in miles
- wind speed - float - mean wind speed for the day in knots
- max sustained wind - float - maximum sustained wind speed for the day in knots
- gust - float - maximum sustained gust
- max temp - float - maximum observed temperature for the day
- min temp - float - minimum observed temperature for the day
- precipitation - float - precipitation for the day in inches
- snow depth - float - depth of snow in inches
- weather events - text - categorical data about weather events extracted from the FRSHTT field in the source data

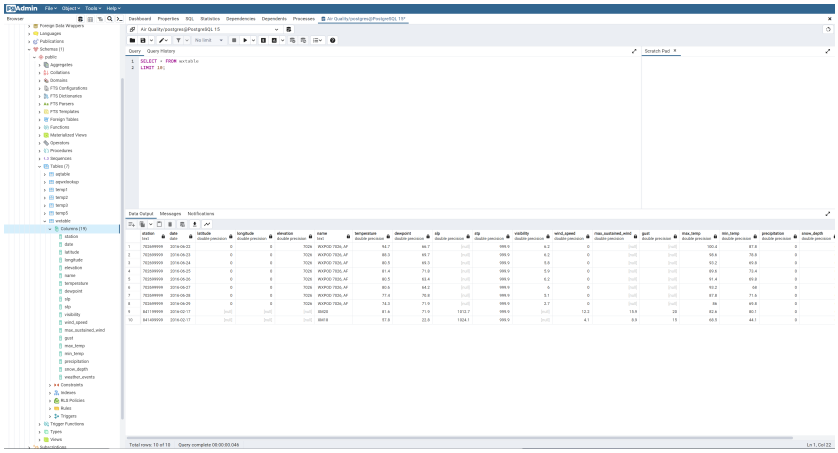

Fig. 8. First 10 rows of wxtable from database.

4 Exploratory Analysis

Having combined the disparate data sources into a locally hosted data warehouse (SQL database) the initial goal of exploration was to get an idea of the geographic distribution of air quality observations and weather stations. This was achieved with a SELECT DISTINCT query in both tables. These queries were then exported into CSV files for analysis with Tableau and use in Jupyter. Query details are below along with visualizations from tableau.

```
SELECT DISTINCT city, "location", latitude, longitude
FROM aqtable
WHERE latitude != 0
ORDER BY city ASC;

SELECT DISTINCT station, "name", latitude, longitude
FROM wxtable
```

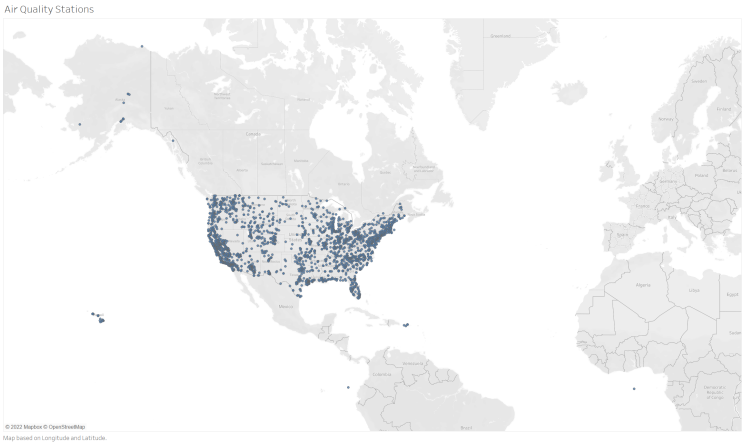

Fig. 9. The geographic distribution of air quality monitoring stations.

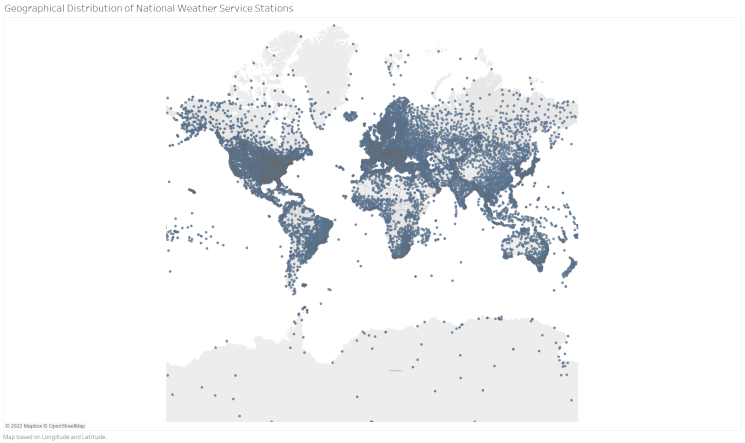

Fig. 10. The geographic distribution of weather monitoring stations.

The S3 bucket for the NOAA Global Surface Summary of the Day actually contained data from monitoring stations around the world. Manual inspection of the data reveals that a few (¡10 points) from the air quality data have erroneous latitudes and longitudes outside of the country. Comparing the highest count cities from the air quality table, there were not exact latitude longitude matches between air quality and weather stations. To utilize these datasets in tandem it was necessary to make a mapping between the air quality and weather stations.

4.1 Making a Lookup Table for Weather and Air Quality Stations

As there are more locations represented by the weather data. The mapping was constructed by matching every air quality station with the closest weather station. The distance was computed using the haversine distance using the haversine library for python [11]. The key logic is detailed in the code snippet below, as the algorithm is O(mn) the execution time can be somewhat lengthy (the haversine function is also somewhat heavy for a tight loop in python). The full details can be seen in the lat long pair generator.ipynb file on the github repository (`https://github.com/matthew-henry/Data-Analytics-Capstone-Project/tree/main/Exploration%20and%20Modeling`). An alternative approach could be made by using a cross join on aqtable and wxtable, but as the GIS(Geographic Information Systems) extensions for PostgreSQL were not installed it was elected to carry out the processing in python.

```
for ind1 in aq_df.index:
    latitude1 = aq_df['latitude'][ind1]
    longitude1 = aq_df['longitude'][ind1]
    coordinate1 = (latitude1, longitude1)
    temp_list=[]
    for ind2 in wx_df.index:
        latitude2 = wx_df['latitude'][ind2]
        longitude2 = wx_df['longitude'][ind2]
        coordinate2 = (latitude2, longitude2)
        distance = hs.haversine(coordinate1, coordinate2)
        temp_list.append((coordinate1, coordinate2, distance))
    distance_list.append(min(temp_list, key=lambda tup: tup[2]))
```

The data was then arranged into a dataframe containing the following fields: aq_city, aq_location, aq_lat, aq_lon, wx_station, wx_name, wx_lat, wx_lon, and distance (description of fields in database below below). This file was exported as a CSV file and imported as a table into the SQL database (aqwxlookup). While the distances between matching stations was available in memory a histogram was plotted and the descriptive statistics computed (see below).

- ind - int - the indicy from the position in the dataframe
- aq city - text - the city field from the air quality table
- aq location - text - the location field from the air quality table
- aq lat - float - the latitude from the air quality table
- aq lon - float - the longitude from the air quality table
- wx station - text - the station from the weather station table
- wx name - text - the name from the weather station table
- wx lat - float - the latitude from the weather station table
- wx lon - float - the longitude from the weather station table
- distance - float - the haversine distance between the paired stations (km)

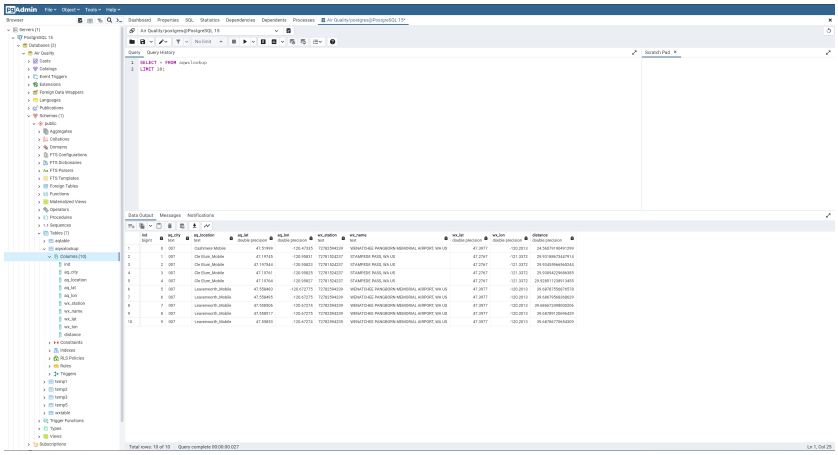

Fig. 11. First 10 rows of the aqwxlookup table in database.

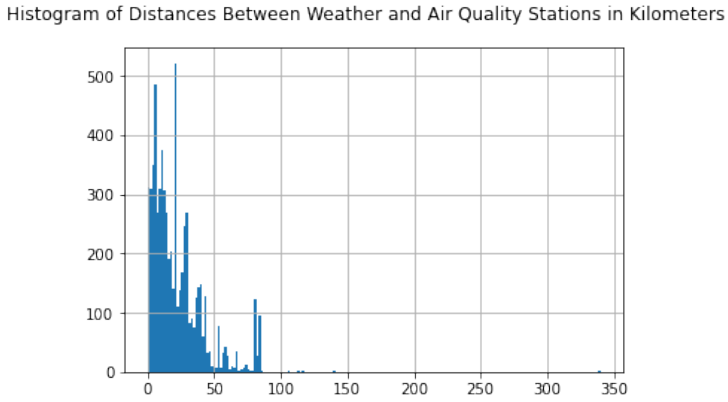

Fig. 12. Histogram of the distances between paired stations.

Count	6436
Mean	22.099115 km
Standard Deviation	19.546868 km
Minimum	0.000000 km
25 percentile	7.730053 km
50 percentile	17.352551 km
75 percentile	29.928896 km
max	340.188868 km

The median separation between paired stations is 17.352551 km, this is larger than desirable for the intended modeling purposes. The lower quartile are within 7.73 km, which is more reasonable; however, this indicates that most of the locations I have data on are not suitable. A SQL query was written to list the top 20 station pairs within 2 miles that have the highest counts of air quality data (the output is included below).

```
SELECT city, COUNT(*)
INTO TEMP TABLE counts
FROM aqtable
GROUP BY city;

SELECT *
FROM aqwxlookup
INNER JOIN counts
ON aqwxlookup.aq_city = counts.city
WHERE distance < 2 AND NOT aq_lat = 0
ORDER BY count DESC
LIMIT 20;
```

	ind bigint	aq_city text	aq_location text	aq_lat double precision	aq_lon double precision	wx_station text	wx_name text	wx_lat double precision	wx_lon double precision	distance double precision	city text	count bigint
1	5462	Phoenix-Mesa-Scottsdale	Eastwood	33.41046	-112.00264	72278023183	PHOENIX AIRPORT, AZ US	33.4277	-112.0069	1.920602394261213302	Phoenix-Mesa-Scottsdale	6349143
2	5463	Phoenix-Mesa-Scottsdale	Falcon Field Fire St	33.4522	-111.7331	72278393185	MESA FALCON FIELD, AZ US	33.46667	-111.73333	1.6091342741924979	Phoenix-Mesa-Scottsdale	6349143
3	5479	Phoenix-Mesa-Scottsdale	Pinal Air Park	32.50831	-111.30806	74916799999	PINAL AIRPARK, AZ US	32.510624	-111.32769	1.8771989270335943	Phoenix-Mesa-Scottsdale	6349143
4	5451	Phoenix-Mesa-Scottsdale	C GRANDE	32.9544	-111.7622	72274803914	CASA GRANDE MUNICIPAL AIRPORT, AZ US	32.95	-111.76667	0.6429077170274864	Phoenix-Mesa-Scottsdale	6349143
5	5464	Phoenix-Mesa-Scottsdale	Falcon Field Fire St	33.4522	-111.7331	72278393185	MESA FALCON FIELD, AZ US	33.46667	-111.73333	1.6091342741924979	Phoenix-Mesa-Scottsdale	6349143
6	4670	Los Angeles-Long Beach-Santa Ana	North Hollywood	34.181979	-118.38304	72288023132	BURBANK GLENDALE PASADENA AIRPORT, CA US	34.19966	-118.36043	1.9786230770446134	Los Angeles-Long Beach-Santa Ana	4697593
7	5672	Riverside-San Bernardino-Ontario	Big Bear	34.264706	-116.86166	A0685400115	BIG BEAR CITY AIRPORT, CA US	34.264	-116.854	0.7082951911450992	Riverside-San Bernardino-Ontario	4550550
8	5701	Riverside-San Bernardino-Ontario	Upland	34.0986	-117.6733	72692999999	CABLE, CA US	34.111562	-117.687569	1.9498128010013807	Riverside-San Bernardino-Ontario	4550550
9	4954	New York-Northern New Jersey-Long Island	Division Street	40.7142	-73.995	72055399999	PORT AUTH DOWNTN MANHATTAN WALL ST HEL, NY US	40.701214	-74.009038	1.8662409491993978	New York-Northern New Jersey-Long Island	3719170
10	4945	New York-Northern New Jersey-Long Island	Babylon	40.7256	-73.4083	74486454747	FARMINGDALE REPUBLIC AIRPORT, NY US	40.73463	-73.41627	0.8775093997449727	New York-Northern New Jersey-Long Island	3719170
11	6329	Washington-Arlington-Alexandria	Ohio Drive	38.87605	-77.03313	99731499999	WASHINGTON, DC US	38.87	-77.02	1.3208185002449846	Washington-Arlington-Alexandria	2839015
12	6323	Washington-Arlington-Alexandria	Frederick Airport	39.42276	-77.37519	72208113730	FREDERICK MUNICIPAL AIRPORT, MD US	39.41667	-77.38333	0.9733887911606055	Washington-Arlington-Alexandria	2839015
13	4583	Las Vegas-Paradise	Indian Springs	36.569443	-115.67667	74614123141	INDIAN SPRINGS, NV US	36.58333	-115.68333	1.6547313313815546	Las Vegas-Paradise	2832482
14	4582	Las Vegas-Paradise	Indian Springs	36.569532	-115.67665	74614123141	INDIAN SPRINGS, NV US	36.58333	-115.68333	1.6668936624108165	Las Vegas-Paradise	2832482
15	5781	Sacramento--Arden-Arcade--Roseville	Davis - UCD Campus	38.53455	-121.7734	72057400174	UNIVERSITY AIRPORT, CA US	38.533	-121.783	0.8526226844931319	Sacramento--Arden-Arcade--Roseville	2812925
16	5441	Philadelphia-Camden-Wilmington	NEA	40.0767	-75.0108	72406594732	NORTHEAST PHILADELPHIA AIRPORT, PA US	40.07692	-75.01336	0.3292096020689945	Philadelphia-Camden-Wilmington	2185614
17	4814	Minneapolis-St. Paul-Bloomington	Blaine	45.1425	-93.2139	72657794974	MINNEAPOLIS ANOKA CO AIRPORT, MN US	45.15	-93.21667	0.8617931828823312	Minneapolis-St. Paul-Bloomington	2093976
18	5828	Salt Lake City	Utah Tech. Center	40.7771	-111.945	72572024127	SALT LAKE CITY INTERNATIONAL AIRPORT, UT US	40.77069	-111.96503	1.8310885586861124	Salt Lake City	1993918
19	5975	Seattle-Tacoma-Bellevue	Seattle-10th & Weller	47.59722	-122.319725	99401499999	SEATTLE, WA US	47.605	-122.338	1.6204569752427125	Seattle-Tacoma-Bellevue	1977949
20	5979	Seattle-Tacoma-Bellevue	Seattle-South Park	47.5297	-122.3203	72793524234	SEATTLE BOEING FIELD, WA US	47.53028	-122.30083	1.4622176383933913	Seattle-Tacoma-Bellevue	1977949

Fig. 13. Results from the above query. A more legible file is hosted in the github repository (`https://github.com/matthew-henry/Data-Analytics-Capstone-Project/blob/main/Visualizations/Station%20Paring%20Query.png`)

From this query the most attractive location to pursue exploratory analysis and modeling appeared to be Los Angeles-Long Beach-Santa Ana. The station

pairing distance was 1.98 km and the location had the second highest count (if counting all the separate Phoenix-Mesa-Scottsdale locations together as a single location). Phoenix had more points but they were split over several stations and those stations mapped to different weather stations which would require more nuanced handling. Given time constraints it was decided to focus in on one location for the purpose of developing a proof of concept modeling approach.

4.2 Exploring the Time Series and Variables

The data for the location was extracted from the database with a SELECT * command with the results filtered by a WHERE clause on the city and latitude and longitude, joining over the air quality and weather tables. There were only three air pollutant species monitored at this station: PM2.5, ozone, and nitrogen dioxide. A separate time series for each species was saved into a CSV file. These CSV files are small enough to be stored on github and are available under the names: "LA pm25.csv", "LA o3.csv", and "LA NO2.csv" (`https://github.com/matthew-henry/Data-Analytics-Capstone-Project/tree/main/Exploration%20and%20Modeling`). All Tableau visualizations were carried out while averaging over the variable of interest (e.g. all observations where the temperature was the same would be averaged together into a single data point). The tableau files can be found on the github repository (`https://github.com/matthew-henry/Data-Analytics-Capstone-Project/tree/main/Visualizations`). It is important to keep in mind that the data presented is being visualized using Tableau's data aggregation features; this improves the visibility of trends, but this increases the R^2 and other measures of fit above what will be actually encountered when regressing over the untransformed data.

The PM2.5 Time Series Upon inspecting the time series for the PM2.5 data there are large peaks that occur every July. These peaks all fall on the day July 5, this implicates July 4 fireworks as a substantial contributer to short term PM2.5 levels. The peak was particularly high in 2020, it would be difficult to investigate, however, whether this would be due to a larger July 4 celebration or favorable weather conditions given that there are only 3 observances of July 4 in the data set. Furthermore, there is a sharp rise in PM2.5 in late September 2020. This may be attributable to the Bobcat Fire which broke out on September 20, 2020 [3]. There does not appear to be a strong seasonality in the data, although due to the high variability any such seasonality could be covered up.

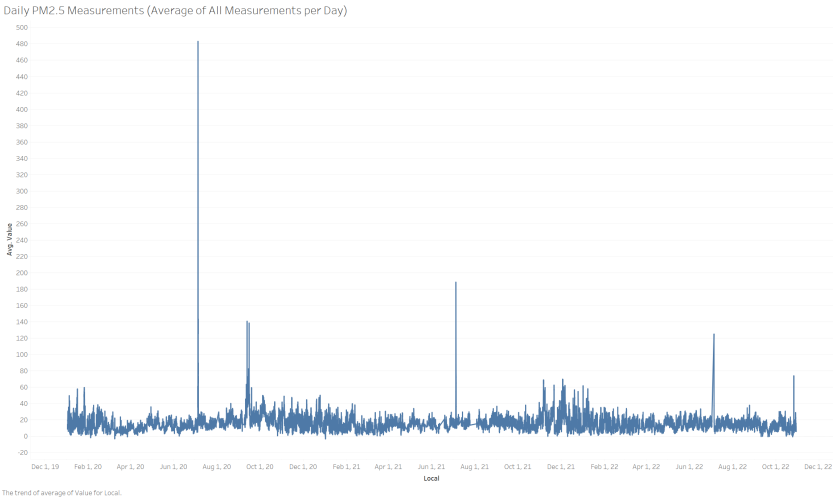

Fig. 14. Tableau Visualization of PM2.5 measurements by day.

The least significant variable is precipitation amount with a P-value of 0.15265. Even though the slope trends downwards there is significant variation and the slope may be driven by outliers on low precipitation days. For precipitation values >0.2 the trend appears relatively flat. The shape of the graph suggested that perhaps an effect might be had by any amount of non-trace precipitation, however, days where rain is observed to occur does not appear to have a different average concentration of PM2.5 than days without weather observation. Precipitation observations are for the day of measurement, and it may be that any particulate scavenging that occurs has a delay that is not captured in this analysis. Despite the weak correlation, some exploration will be invested into determining whether a non-linear relationship with precipitation might be predictive.

Dewpoint and minimum temperature appear to have very modest impacts on PM2.5 concentrations. The p-value for minimum temperature is 0.0882657, but it may be prudent to leave it in for modeling as other temperature inputs appear to be more significant. Dewpoint is of a similar correlation but has a significant P-value. Both variables are slightly positively correlated with PM2.5 concentration. Both variables will be included in modeling.

Station pressure has a modest negative correlation with PM2.5 concentrations. It is not clear if this is a function of high pressure systems on wind and temperature or an inherent result of the barometric pressure. The value is significant, and will included in modeling.

Maximum temperature and temperature have positive correlations with PM2.5 concentrations, the correlation for maximum temperature, however, is much stronger. It appears that the most significant temperature related factor for

PM2.5 is the high temperature reached for the day, with low temperatures having a negligible impact and weakenening the correlation of the average temperature.

All wind parameters have strong negative correlations with PM2.5, and it appears that the presence of a strong wind is the primary factor in clearing PM2.5. The most strongly correlated wind parameter is maximum sustained wind speed, and the weakest is the maximum gust. There are many parameters where gust is null, so it will not be included in modeling efforts.

Visibility has a strong negative correlation. However, given that there is reason to suspect that PM2.5 concentrations contribute to low visibility and that air quality metrics are used to forecast visibility, the variable was left out of modeling efforts.

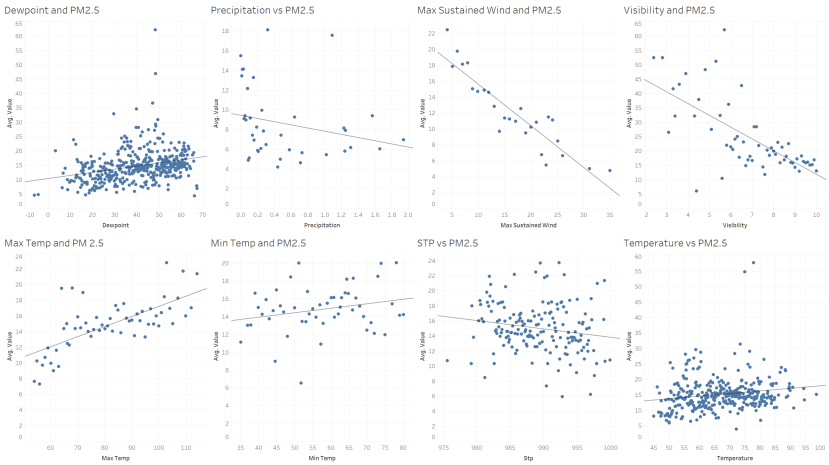

Fig. 15. Dashboard Containing Scatter Plots for PM2.5 and Weather Data

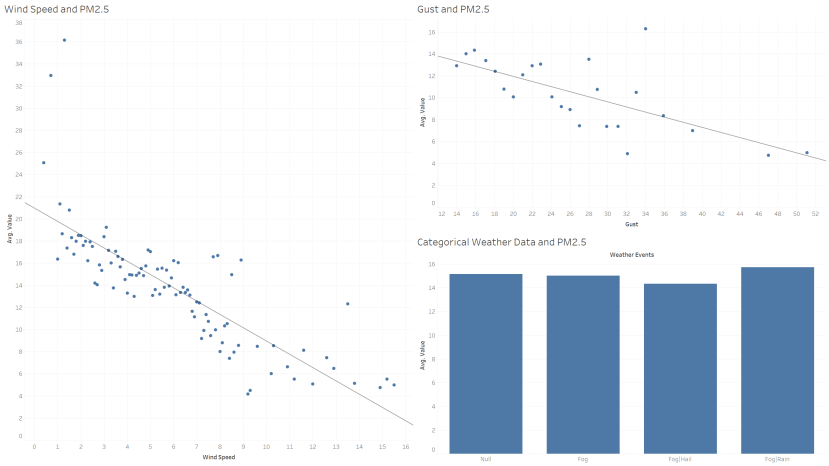

Fig. 16. Dashboard Containing Scatter Plots and Bar Charts for PM2.5 and Weather Data

Variable	R^2	P-Value
Dewpoint	0.0840692	< 0.0001
Precipitation	0.0559838	0.15265
Max Sustained Wind	0.83373	< 0.0001
Wind Speed	0.652347	< 0.0001
Gust	0.474448	0.0001405
Visibility	0.461286	< 0.0001
Maximum Temperature	0.464519	< 0.0001
Minimum Temperature	0.0570062	0.0882657
Temperature	0.0300663	0.0012638
Station Pressure	0.0323363	0.0189523

The Ozone Time Series The ozone time series is fairly noisy, but appears to be roughly stationary with a seasonal component. Values are elevated from June to September and lower during the colder months.

Station pressure has a significant negative correlation with ozone concentrations. Unlike with PM2.5, station pressure will likely be a very significant variable for forecasting ozone.

As with PM2.5 Ozone has significant positive correlations with temperature variables. The relationships appear fairly linear but may be trending slightly non-linear on the upper end of the temperature range.

In a complete reversal from PM2.5's trends Ozone concentrations correlate positively with wind speed variables. Once again the correlation is strongest with max sustained and average wind speed in comparison to gust. As gust has nulls in all time series, it will be omitted from ozone modeling as well.

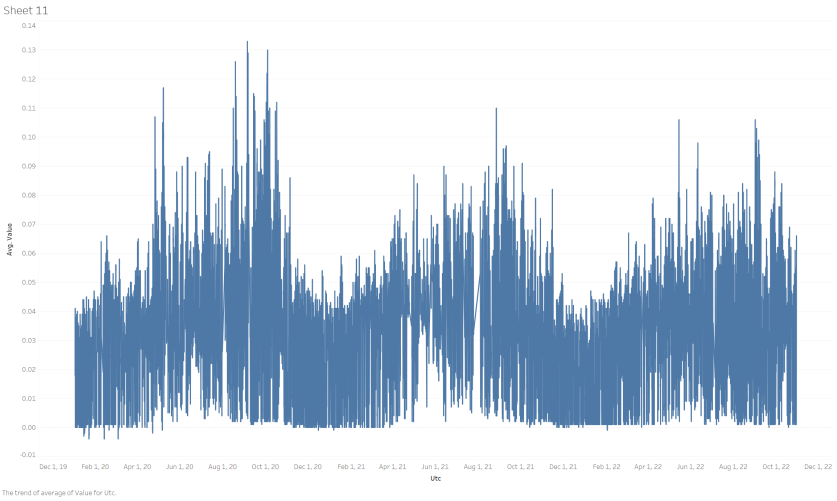

Fig. 17. Tableau Visualization of Ozone measurements by day.

While days with reported rain do appear to have a lower recorded ozone value, there is no significant correlation between ozone and precipitation amounts. As such like with PM2.5, precipitation will be investigated for a non-linear response in preliminary modeling.

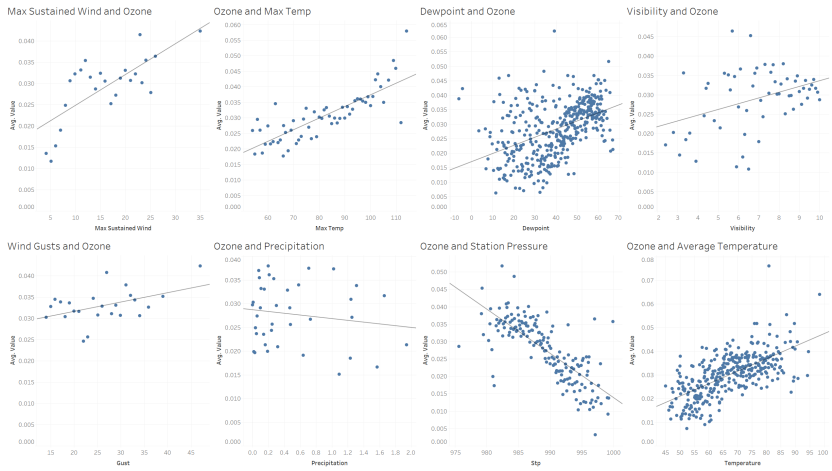

Fig. 18. Dashboard Containing Scatter Plots for Ozone and Weather Data

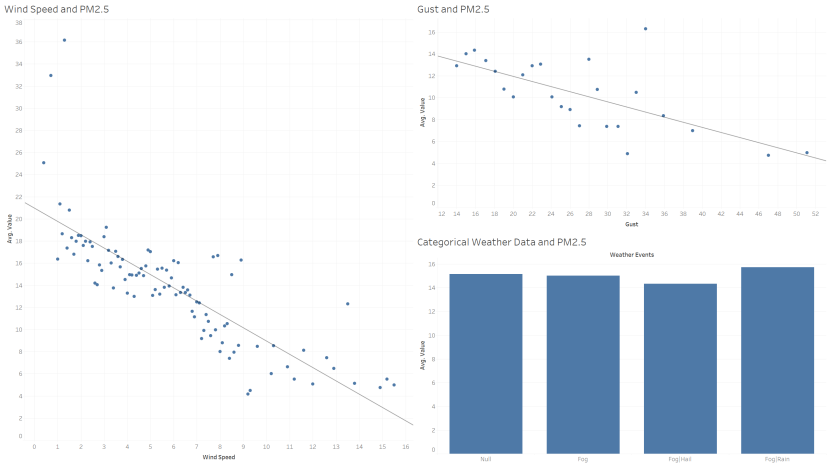

Fig. 19. Dashboard Containing Scatter Plots and Bar Charts for Ozone and Weather Data

Variable	R^2	P-Value
Dewpoint	0.206647	< 0.0001
Precipitation	0.0236006	0.367126
Max Sustained Wind	0.537537	< 0.0001
Wind Speed	0.629037	< 0.0001
Gust	0.214442	0.022677
Visibility	0.149289	< 0.0001
Maximum Temperature	0.653613	< 0.0001
Temperature	0.414046	< 0.0001
Station Pressure	0.572467	< 0.0001

The Nitrogen Dioxide Time Series The time series for the nitrogen dioxide measurements is also stationary with a seasonal component. The seasonal pattern is reversed from the Ozone patterns, in that the nitrogen dioxide values are highest in late fall and winter and early spring. Given that there is a weak positive correlation with maximum temperature (not significant at 95% confidence as P=0.075854) and a weak negative correlation with average temperature, this effect is possibly driven by the dewpoint's strong negative correlation.

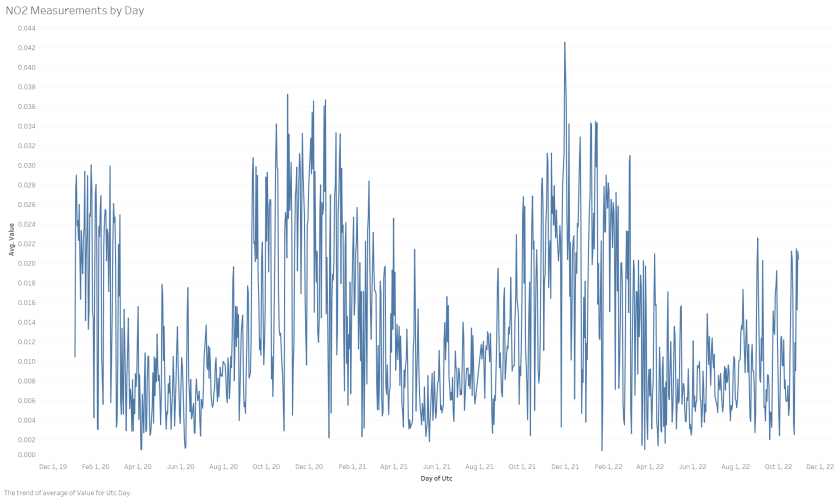

Fig. 20. Tableau Visualization of Nitrogen Dioxide measurements by day.

Despite the strong dewpoint correlation there appears to be no correlation between measured precipitation and nitrogen dioxide concentrations. It is not clear why this should be the case, but nevertheless just as in the other cases precipitation does not appear likely to have explanatory or predictive value.

As with PM2.5, nitrogen dioxide shows a strong negative correlation with wind variables, with gust once again being the weakest (with P¿0.05). There is a positive correlation between measured atmospheric pressure and NO2, this is inverted from Ozone's behavior.

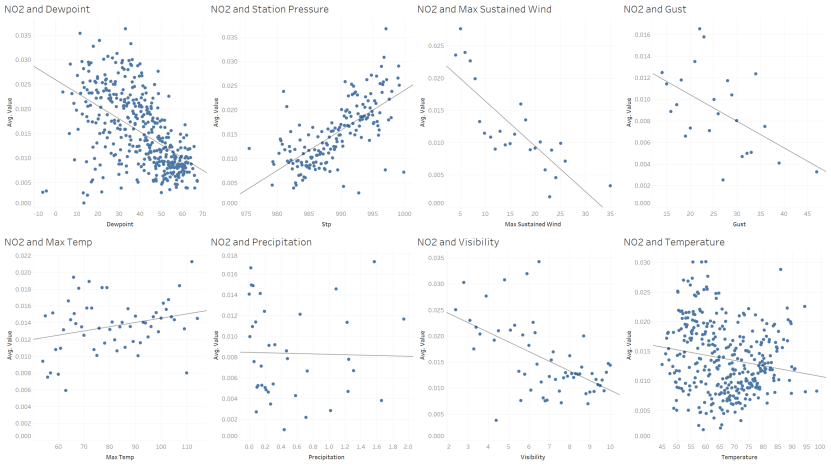

Fig. 21. Dashboard Containing Scatter Plots for Nitrogen Dioxide and Weather Data

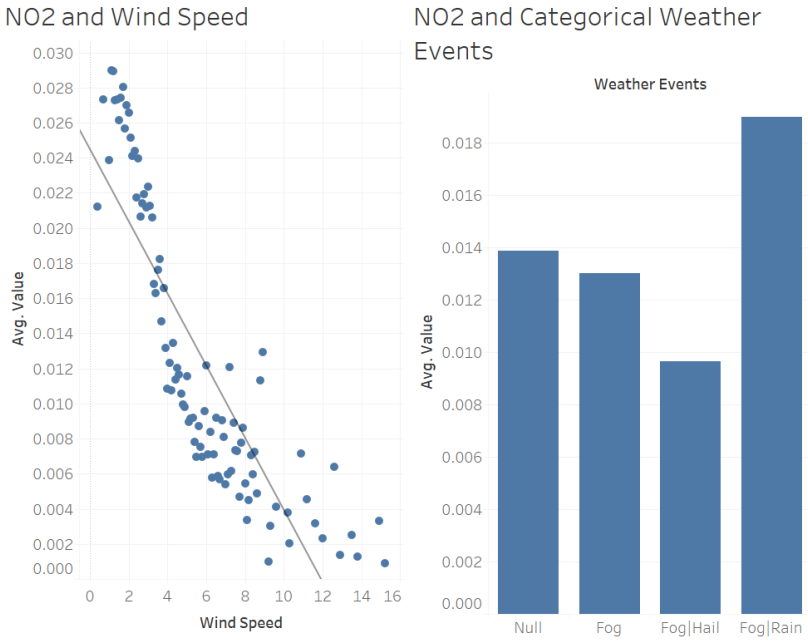

Fig. 22. Dashboard Containing Scatter Plots and Bar Charts for Nitrogen Dioxide and Weather Data

Variable	R^2	P-Value
Dewpoint	0.289575	< 0.0001
Precipitation	0.0005343	0.890435
Max Sustained Wind	0.662227	< 0.0001
Wind Speed	0.741783	< 0.0001
Gust	0.2282156	0.007564
Visibility	0.149289	< 0.0001
Maximum Temperature	0.653613	< 0.0001
Temperature	0.414046	< 0.0001
Station Pressure	0.572467	< 0.0001

5 Predictive Modeling

Having determined variables of interest, three modeling approaches were pursued: a simple exploratory regression approach using linear and non-linear regression models, an approach using autoregressive time series approaches, and the deployment of an LSTM neural network. Of these approaches the regression approach should be the weakest as it is not a method intended for treating time series, however, given the relatively robust correlations observed during the exploratory analysis there is reason to believe that there might be some utility in such a modeling approach. The regression results will also provide insight into how exogenous variables might be handled in an autoregressive model.

5.1 Preprocessing for Modeling

In each model building notebook the data was preprocessed by resampling the data to a daily grain and interpolating missing potes using pandas time-series interpolation method. The sampling to a daily grain was done as even though air quality was often gathered multiple times a day, the intervals were inconsistent and the weather data was only tabulated on a daily basis. The general format of the transformation of the dataframe for this preprocessing is provided below. The data was split into a test train split using sklearn's train test split module, 20% of the data set was withheld for the test set.

```
df = pd.read_csv('filename.csv', parse_dates=['date', 'utc','local'], index_col='utc')
df = df.resample('d').mean()
df = df.interpolate(method='time')
```

5.2 Regression Models

For each of the species in question a linear regression model (of polynomial degree 1,2, and 3), a random forest regression, and a multilayer perceptron regression were attempted. Hyper-parameters were selected empirically for the random forest and MLP(Multi-layer Perceptron) models which were not optimized further as they irreconcilably overfit the data. MSE(Mean Squared Error) and R^2 were presented for all regression models.

Linear Regression on PM 2.5 The variables regressed were: station atmospheric pressure, average wind speed, maximum sustained wind speed, maximum temperature, minimum temperature, and dewpoint. A few models with reduced variable count and with precipitation added in were tried, but were not effective in improving model performance and have been withheld from this write-up for brevity. Not interaction terms were included in the polynomial models. Full details can be found in the PM25.ipynb notebook on github (`https://github.com/matthew-henry/Data-Analytics-Capstone-Project/blob/main/Exploration%20and%20Modeling/PM25.ipynb`).

Model Description	Train MSE	Train R^2	Test MSE	Test R^2
Degree 1	47.4272	0.1944	Not Evaluated	Not Evaluated
Degree 2	44.0311	0.25208	25.5734	0.3057
Degree 3	39.1529	0.3349	32.4985	0.1178

At degree 3 the model clearly begins to overfit as evidenced by the drop in R^2 in the test set. The degree two model performs reasonably well for a simple model, typical values are around 20 μg for PM2.5 putting the RMSE(Root Mean Square Error) on the order of 25% of typical values. The R^2 indicates that this simple weather model only explains around 25% of the variance in PM2.5 concentrations, which does seem reasonable given that forest fires and fireworks appear to have caused the largest peaks of PM2.5.

Linear Regression on Ozone The polynomial regression models begin to overfit at degree 3. The degree 2 model explains on the order of 50% of the variation in ozone using meteorological variation. The RMSE is around 0.001 ppm with values for ozone ranging around 0.00 to 0.10 ppm.

Model Description	Train MSE	Train R^2	Test MSE	Test R^2
Degree 1	$7.5534*10^-5$	0.4513	Not Evaluated	Not Evaluated
Degree 2	$6.48*10^-5$	0.5292	$7.8926*10^-5$	0.4736
Degree 3	$5.4196*10^-5$	0.6063	$8.9017*10^-5$	0.4064

Linear Regression on Nitrogen Dioxide The regression models for Nitrogen Dioxide have very good performance. Overfitting begins at degree 3. R^2 for the degree 2 model is on the order of 70% indicating that the majority of the variation in nitrogen dioxide concentrations is explainable by weather variables. The data varies from 0 to 0.14 ppm with a model RMSE of 0.004.

Random Forest Regression on PM2.5 The random forest regressor overfit the data in all evaluated circumstances. While reducing the number of estimators

Model Description	Train MSE	Train R^2	Test MSE	Test R^2
Degree 1	$2.0912*10^-5$	0.6886	Not Evaluated	Not Evaluated
Degree 2	$1.1721*10^-5$	0.7437	$1.9997*10^-5$	0.7047
Degree 3	$1.2892*10^-5$	0.8081	$2.7171*10^-5$	0.5984

and the maximum tree depth did reduce the degree of overfitting the quality of the model was deteriorating to the point it was approaching comparable to the degree 2 regression model while still being overfit. As such the random forest regressor was not considered appropriate for modeling.

Model Description	Train MSE	Train R^2	Test MSE	Test R^2
estimators=1000 max depth=100	6.2842	0.8933	27.8786	0.2432
estimators=100 max depth=10	11.6902	0.8014	27.5573	0.2519
estimators=10 max depth=10	14.0442	0.7614	31.6915	0.1397
estimators=50 max depth=5	32.8226	0.4425	28.4900	0.2266

Random Forest Regression on Ozone By reducing the max tree depth it is possible to generate a random forest regression model that is not overfit. However, the performance of the model is comparable the polynomial regression model when not overfit.

Model Description	Train MSE	Train R^2	Test MSE	Test R^2
estimators=100 max depth=100	$9.000*10^-6$	0.9346	$7.6853*10^-5$	0.4875
estimators=100 max depth=50	$9.4803*10^-6$	0.9311	$7.7557*10^-5$	0.4828
estimators=100 max depth=15	$1.1202*10^-5$	0.9186	$7.7560*10^-5$	0.4959
estimators=100 max depth=10	$2.1041*10^-5$	0.8472	$7.7561*10^-5$	0.4958
estimators=100 max depth=5	$5.1746*10^-5$	0.6241	$7.7524*10^-5$	0.4983
estimators=100 max depth=3	$6.6272*10^-5$	0.5186	$7.7985*10^-5$	0.4766

Random Forest Regression on Nitrogen Dioxide The random forest model was still overfitting as it's performance on the train set approached the degree 2 regression model. As such the optimization of the hyper parameters was stopped.

MLP Regression on PM2.5 The MLP regressor also had a tendency to overfit the data, and adjusting the hyperparameters to reduce overfitting led to a model worse than that achievable by linear regression.

Model Description	Train MSE	Train R^2	Test MSE	Test R^2
estimators=100 max depth=100	$2.5691*10^-6$	0.9617	$2.0841*10^-5$	0.6923
estimators=100 max depth=50	$2.6581*10^-6$	0.9604	$7.0840*10^-5$	0.6923
estimators=100 max depth=25	$2.6599*10^-6$	0.9604	$2.0868*10^-5$	0.6919
estimators=100 max depth=10	$4.3782*10^-6$	0.9348	$2.1192*10^-5$	0.6871
estimators=100 max depth=5	$1.4635*10^-5$	0.7821	$2.3630*10^-5$	0.6511
estimators=10 max depth=5	$1.5287*10^-5$	0.7724	$2.4172*10^-5$	0.6431

Model Description	Train MSE	Train R^2	Test MSE	Test R^2
Standard Scaler, layers=(100,400,100), activation=tanh	10.8064	0.8164	36.1357	0.0190
Standard Scaler, layers=(25,50,20), activation=tanh	15.0240	0.7447	30.6984	0.1666
Standard Scaler, layers=(50,100,50), activation=tanh	50.1424	0.1423	30.7588	0.1650

MLP Regression on Ozone The MLP models failed to model the ozone concentrations with all attempts yielding R^2 of near 0. This behavior was unchanged with and without the use of a scaler.

MLP Regression on Nitrogen Dioxide The MLP models failed to model the nitrogen dioxide concentrations. All R^2 values were negative. This behavior was unchanged with and without the use of a scaler.

5.3 Autoregressive Time Series Models

An ARX(n) type model was developed for each of the three pollutants. The models were all validated using backtesting, where the model is shown past data, asked to predict the next point, and it's accuracy scored. The backtesting process is continued throughout the entire dataset. Normally a test or validation set is not withheld when using backtesting, but I chose to do a test train split to evaluate whether the parameters chosen for the model would work correctly outside of the data on which the model was trained. The skforecast library was used to perform this modeling [10]. For each model the chosen regression pipeline was the linear regression model. A linear and second degree model was tested for each, but the second degree model behaved poorly and erratically and is not included in this summary. The process is documented in the: "Time Series P25.ipynb", "Time Series O3.ipynb", and "Time Series NO2.ipynb" files in the github repository (`https://github.com/matthew-henry/Data-Analytics-Capstone-Project/tree/main/Exploration%20and%20Modeling`). Sample code for a modeling setup is included below.

```
X = train_df[['stp', 'wind_speed', 'max_sustained_wind', 'max_temp', 'min_temp',
'temperature', 'dewpoint']]
forecaster = ForecasterAutoreg(regressor = LinearRegression(), lags = 5)
forecaster.fit(y=train_df['value'])

metrics, predictions_backtest = backtesting_forecaster(forecaster = forecaster,
```

```
 y = train_df['value'],
 exog=X,
 initial_train_size = 20,
 fixed_train_size = False,
 steps = 1,
metric = 'mean_squared_error',
 refit = True,
 verbose = True)
```

PM2.5 Autoregressive Model The time series shows significant autocorrelation for about 6 steps back in the series. This suggests that an autoregression model should incorporate 5 or less lags. As such it was elected to implement an ARX(5) model predicting 1 step out (e.g. data from the last 5 days will be used to predict 24 hours into the future).

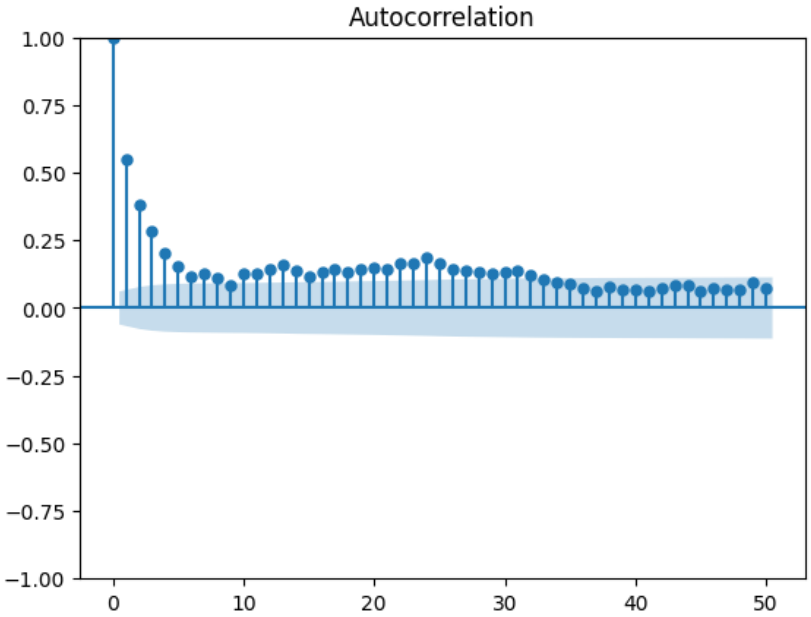

Fig. 23. Autocorrelation plot of PM2.5 Time Series

Training the model it had a MSE of 46.98 on the train set, and a MSE of 12.47 on the test set. Given that there was a major wild fire increasing the noise of the train set this performance seems sensible. The outputs of the model are plotted below, and accurately recapture the behavior of the time series. The model does, however, tend to underestimate how high spikes in PM2.5 go (this

behavior is best illustrated by extremely elevated levels such as was encountered around July 4, 2020 in the graphs below).

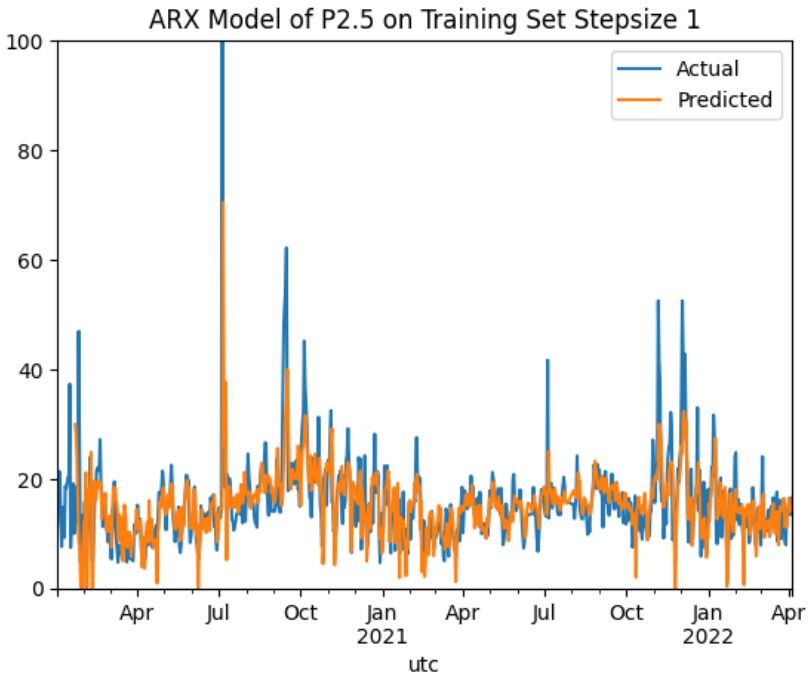

Fig. 24. Performance of autoregressive model on PM2.5 Train Set

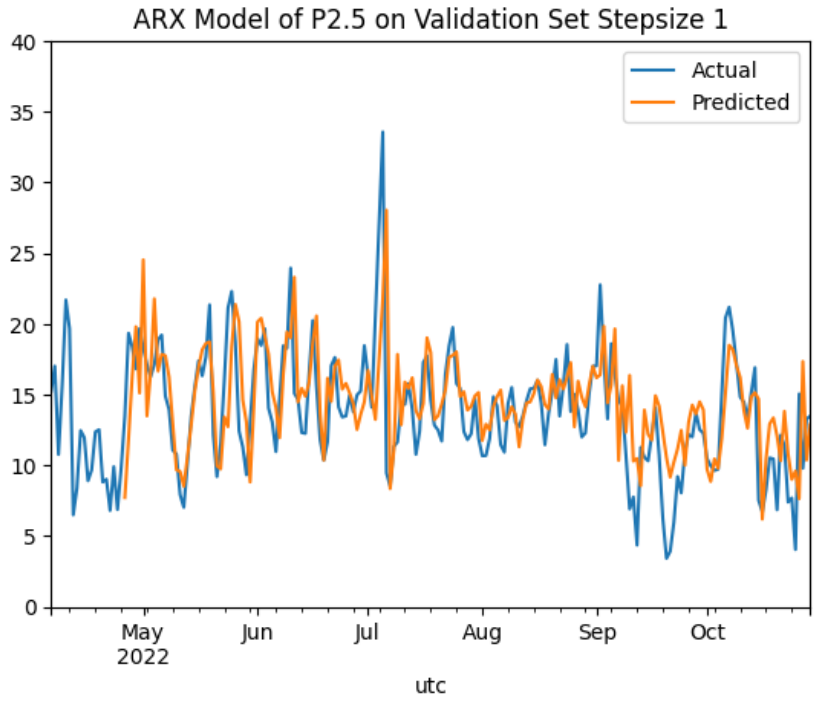

Fig. 25. Performance of autoregressive model on PM2.5 Test Set

Ozone Autoregressive Model The time series shows significant autocorrelation for almost 50 steps back in the series. However, the strongest autocorrelation is in the first 3 to 5 steps. As such it was decided to maintain the 5 lags and 1 step forward approach used in the PM2.5 model.

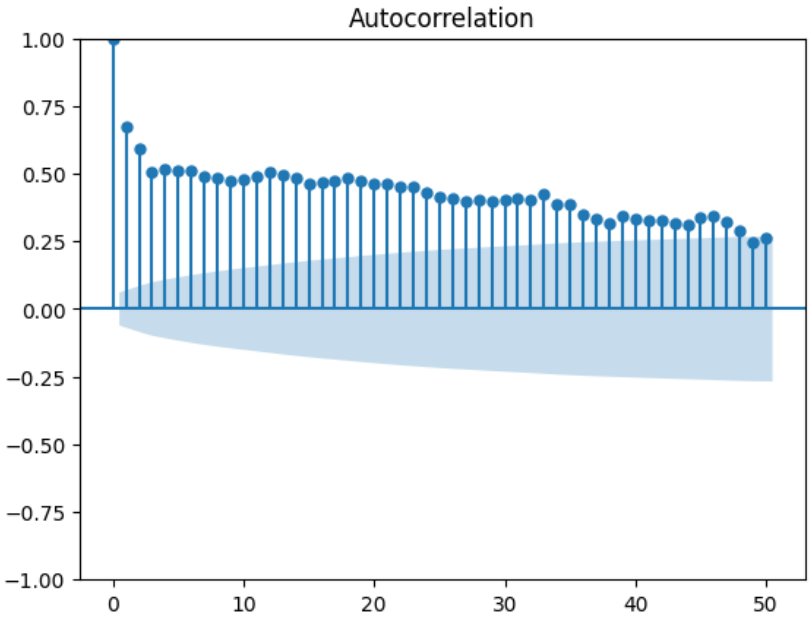

Fig. 26. Autocorrelation plot of Ozone Time Series

The train set had an MSE of $6.618*10^{-}6$ and the test set an MSE of $3.0306*10^{-}5$. Overall the backtesting shows a strong following of the overall trend, although once again the model tends to have lower peaks. Despite not building a seasonal model, the exogenous meteorological variables are sufficient to predict and reproduce the seasonal pattern.

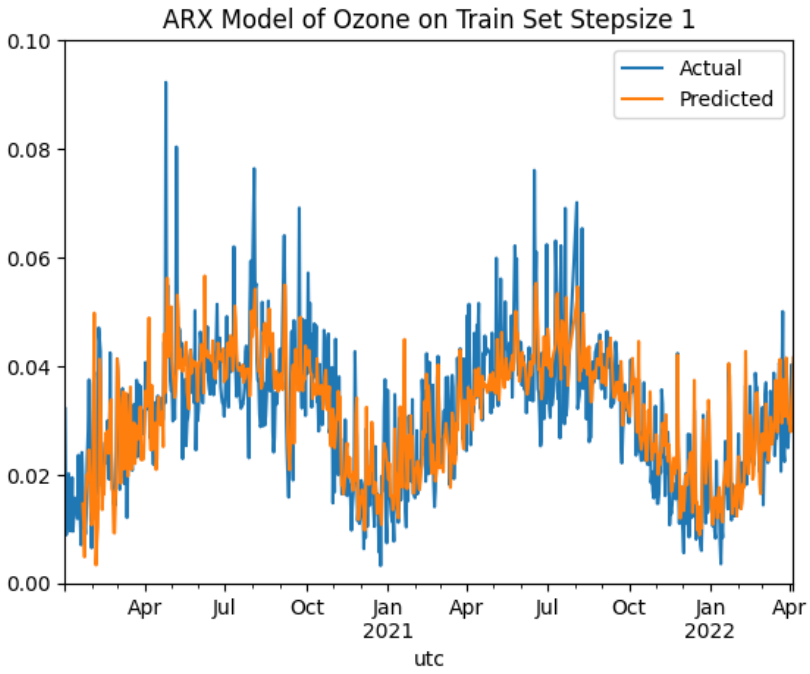

Fig. 27. Performance of autoregressive model on the Ozone Train Set

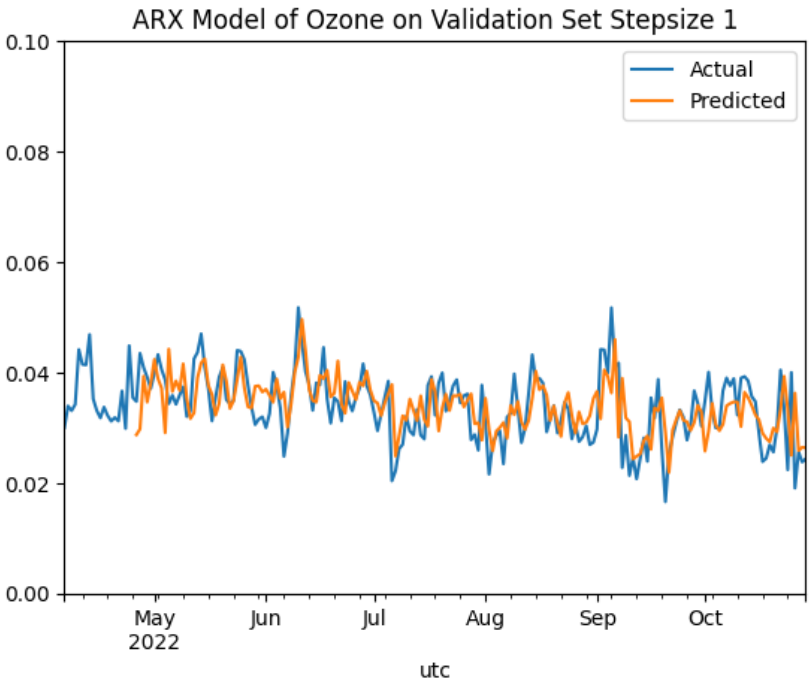

Fig. 28. Performance of autoregressive model on the Ozone Test Set

The Nitrogen Dioxide Time Series The time series shows significant autocorrelation for almost 50 steps back in the series. However, the strongest autocorrelation is in the first 3 to 5 steps. As such it was decided to maintain the 5 lags and 1 step forward approach used in the other models.

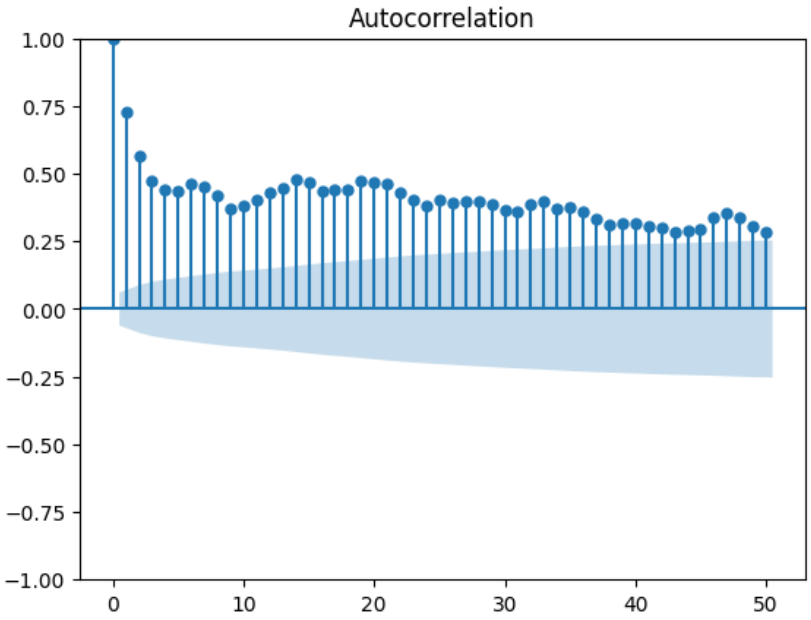

Fig. 29. Autocorrelation plot of Nitrogen Dioxide Time Series

$1.645*10^{-}6$ $7.095*10^{-}6$ The train set had an MSE of $1.645*10^{-}6$ and the test set an MSE of $7.095*10^{-}6$. Overall the backtesting shows a strong following of the overall trend, with the nitrogen dioxide model showing a better following of the peaks than the other two models. Despite not building a seasonal model, the exogenous meteorological variables are sufficient to predict and reproduce the seasonal pattern.

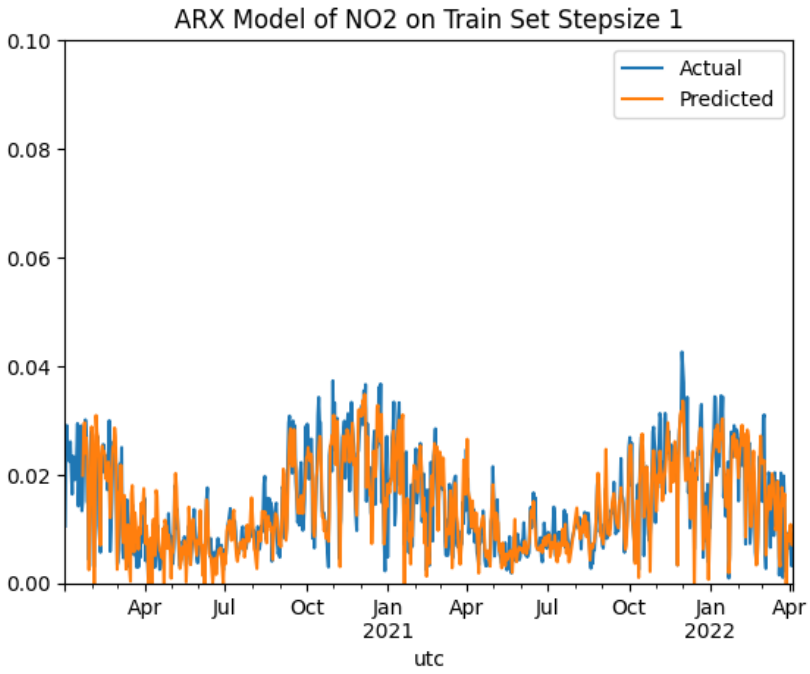

Fig. 30. Performance of autoregressive model on the nitrogen dioxide Train Set

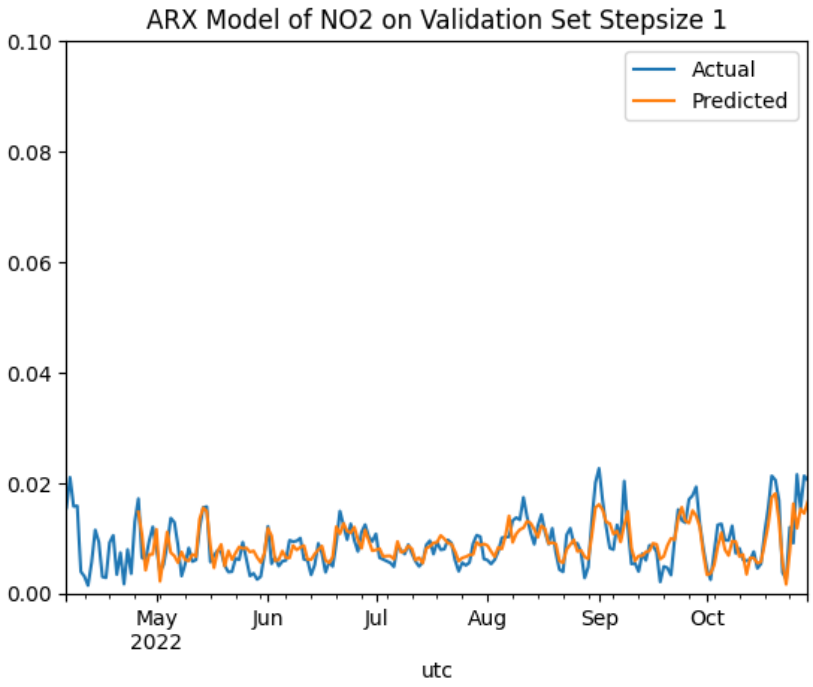

Fig. 31. Performance of autoregressive model on the nitrogen nioxide Test Set

5.4 LSTM Modeling

Due to time constraints there was not much time to explore the neural network's design. The model was fed a window width of 5 datapoints for past concentrations. Before training the data was passed through sklearn's minmaxscaler. Having a separate input layer before the LSTM layers performed worse than going straight into an LSTM layer. The overall design ended up being 64 LSTM neurons feeding into 8 dense neurons with a ReLU activation function feeding into 1 dense neuron. The ReLU activation function showed better training and convergence properties than a sigmoid function. The model was built and trained using keras with a template as follows in the code sample below. The loss curves were plotted and the mean squared error computed. The unscaled outputs were plotted against the train and test data. The notebooks can be found on github as: "PM2.5 neural net.ipynb", "O3 Neural Net.ipynb", and "NO2 Neural Net.ipynb" (`https://github.com/matthew-henry/Data-Analytics-Capstone-Project/tree/main/Exploration%20and%20Modeling`).

```
model = Sequential()
model.add(LSTM(64, input_shape=(5,8)))
model.add(Dense(8, 'relu'))
model.add(Dense(1, 'linear'))
model.compile(loss='mse', optimizer='adam')
model.fit(X, y, epochs=100, validation_data=(valX, valy), verbose=2)
```

LSTM Modeling of PM2.5 The model had a MSE of 37.646 on the training set and a MSE of 23.589 on the test set. As with the time series model the test set is cleaner than the train set so this yields a lower error. The model's qualitative performance looks fairly good on the training set, but performances substantially worse on the test set than the autoregressive model (see graph of actual vs predicted outputs). It is possible that the performance could be improved by altering the design of the neural network, but due to time constraints this path was not pursued.

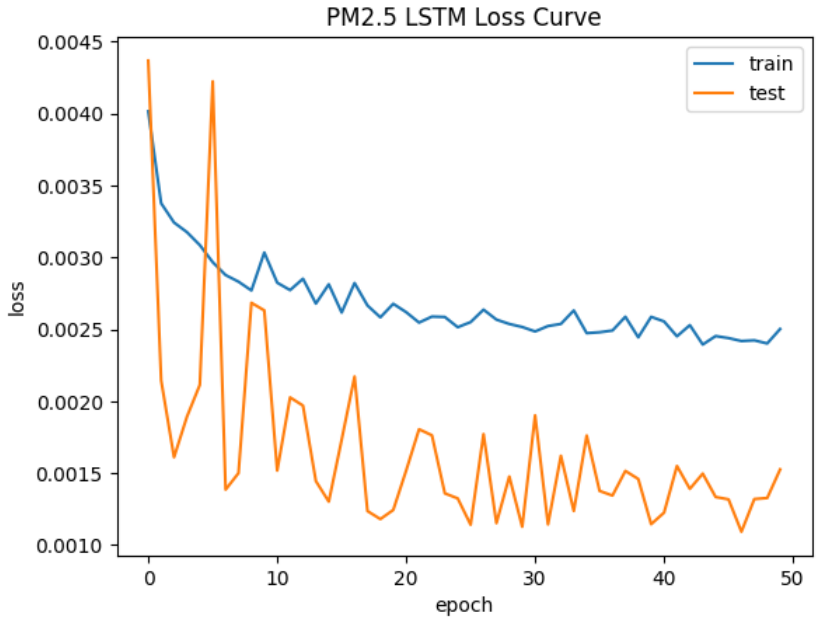

Fig. 32. Loss curve for training of PM2.5 LSTM neural network

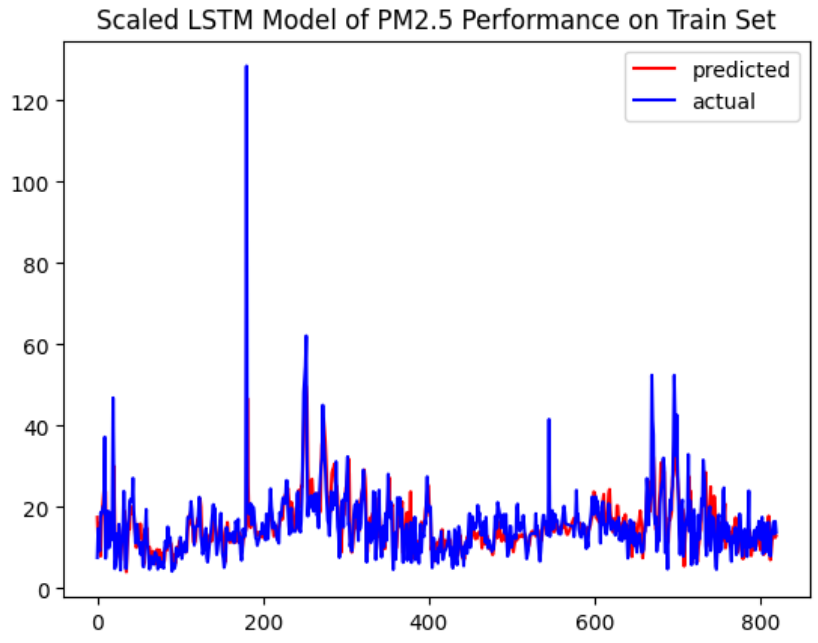

Fig. 33. Performance of the LSTM model on the training set for PM2.5

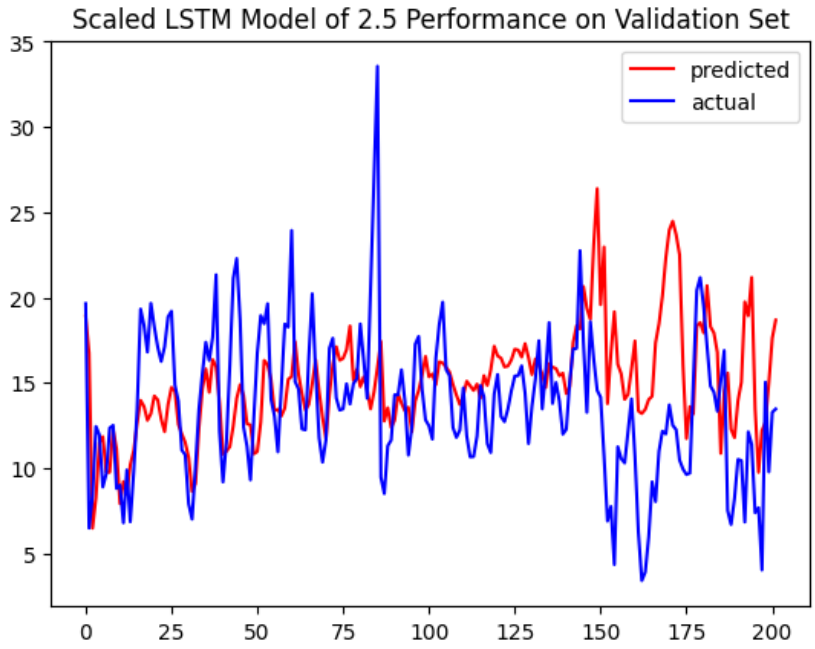

Fig. 34. Performance of the LSTM model on the test set for PM2.5

LSTM Modeling of Ozone The model had a MSE of $7.1147*10^{-}5$ on the training set and a MSE of $3.1147*10^{-}5$ on the test set. As with the time series model the test set is cleaner than the train set so this yields a lower error. The model's qualitative performance looks fairly good on the training set, but performances substantially worse on the test set than the autoregressive model, although appears to generalize better than the PM2.5 model. It is possible that the performance could be improved by altering the design of the neural network, but due to time constraints this path was not pursued.

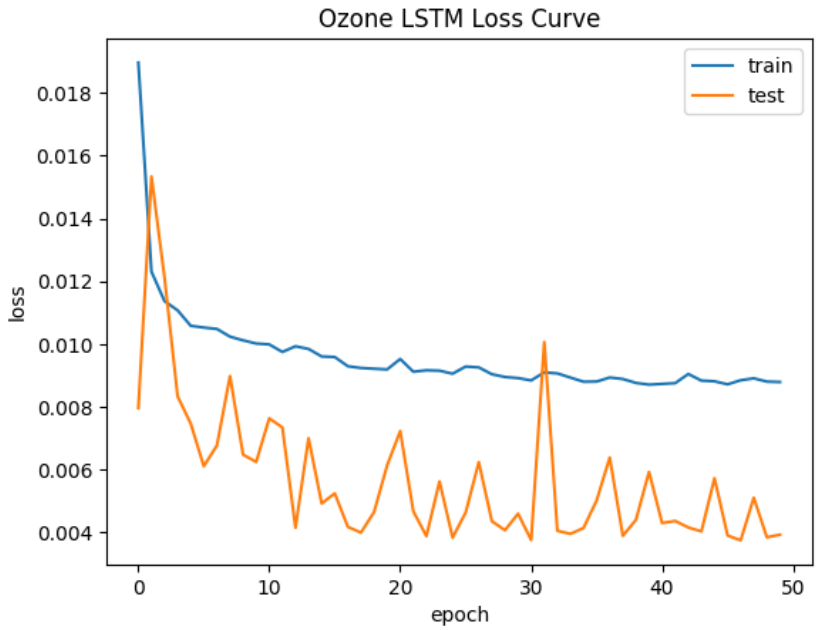

Fig. 35. Loss curve for training of Ozone LSTM neural network

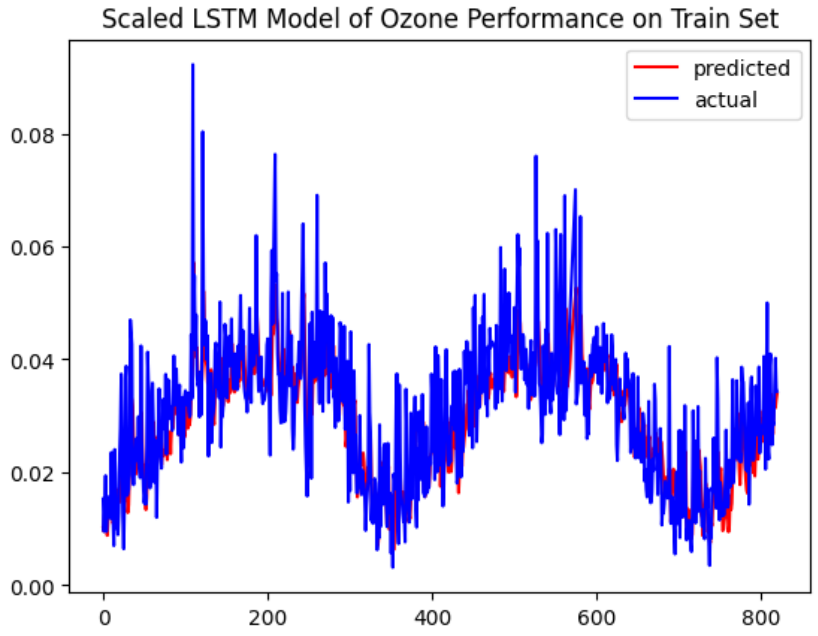

Fig. 36. Performance of the LSTM model on the training set for Ozone

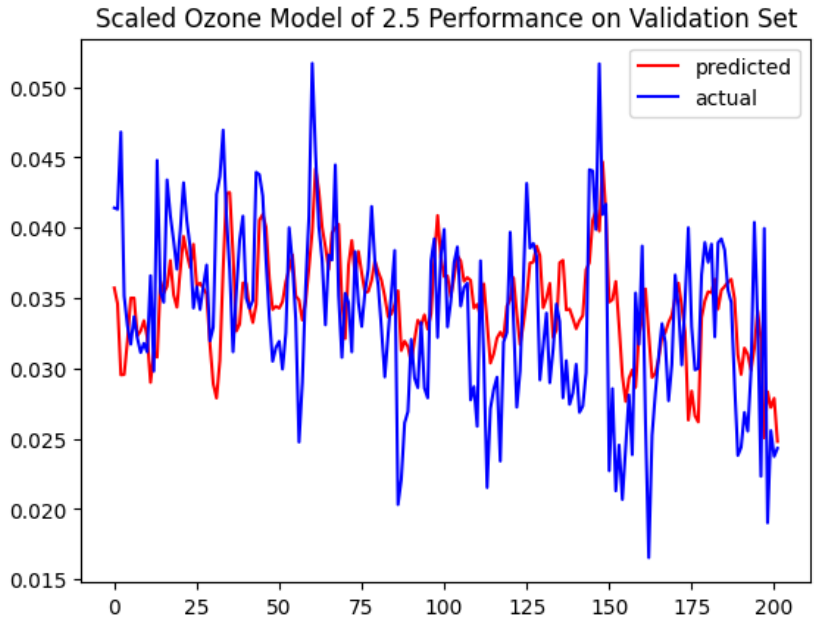

Fig. 37. Performance of the LSTM model on the test set for Ozone

LSTM Modeling of Nitrogen Dioxide The model had a MSE of $2.3767*10^-5$ on the training set and a MSE of $1.0433*10^-5$ on the test set. As with the time series model the test set is cleaner than the train set so this yields a lower error. The model's qualitative performance looks fairly good on the training set, but performances slightly worse on the test set than the autoregressive model(see the graphs of model vs actual output presented below). It is possible that the performance could be improved by altering the design of the neural network, but due to time constraints this path was not pursued.

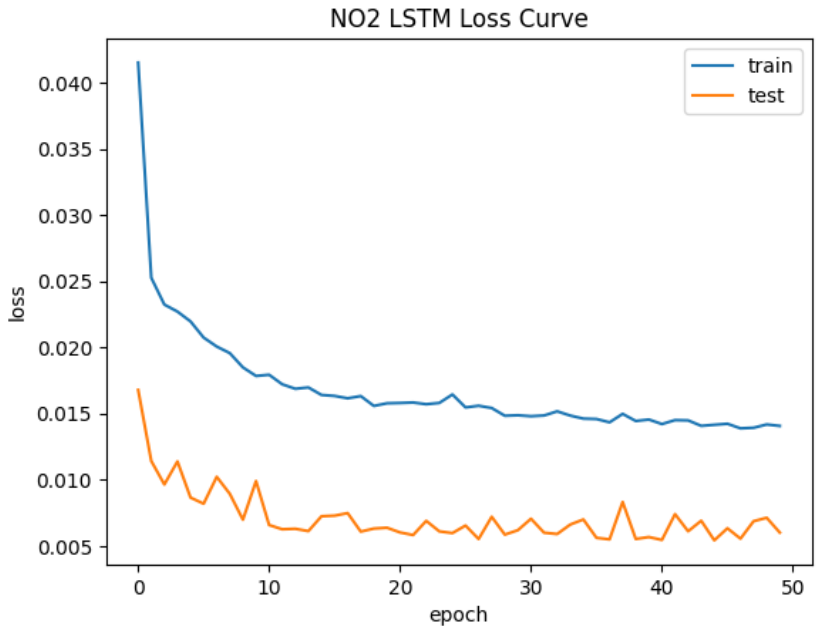

Fig. 38. Loss curve for training of nitrogen dioxide LSTM neural network

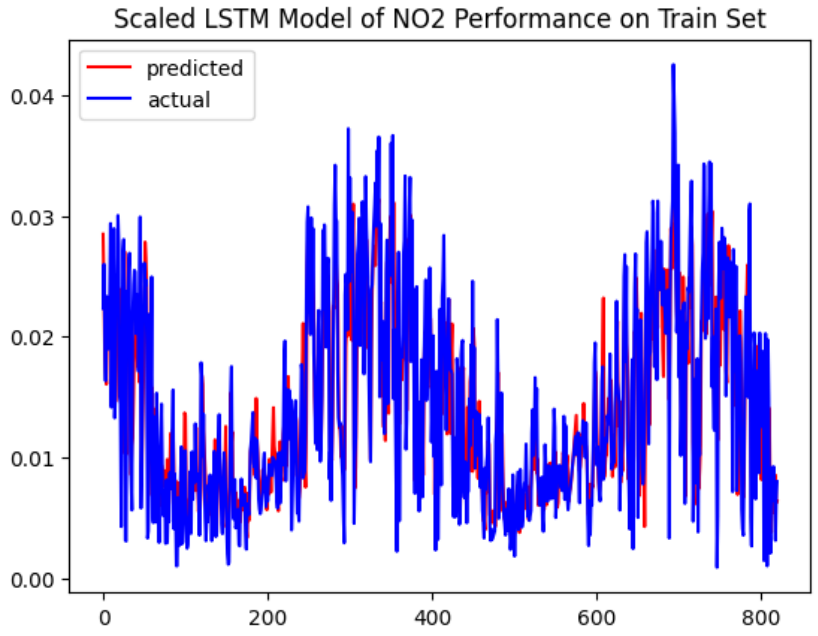

Fig. 39. Performance of the LSTM model on the training set for nitrogen dioxide

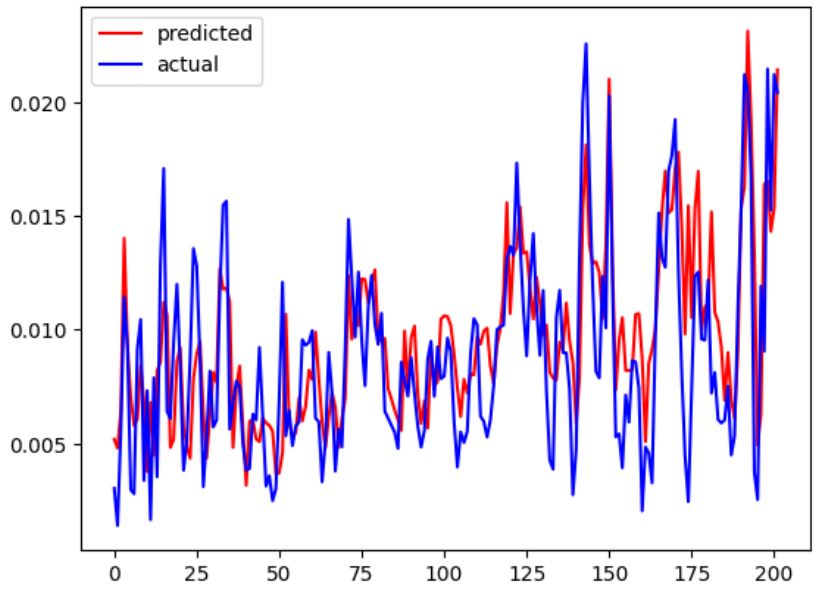

Fig. 40. Performance of the LSTM model on the test set for nitrogen dioxide

6 Results and Analysis

6.1 Summary of Model Performance for PM2.5

Model	Train MSE	Test MSE
Degree 2 Polynomial Regression	44.0311	25.5734
ARX(5)	46.98	12.47
LSTM	37.646	23.589

6.2 Summary of Model Performance for Ozone

Model	Train MSE	Test MSE
Degree 2 Polynomial Regression	$6.480*10^-5$	$7.789*10^-5$
ARX(5)	$6.618*10^-5$	$3.031*10^-5$
LSTM	$7.115*10^-5$	$3.115*10^-5$

6.3 Summary of Model Performance for Nitrogen Dioxide

Model	Train MSE	Test MSE
Degree 2 Polynomial Regression	$1.172*10^-5$	$2.000*10^-5$
ARX(5)	$1.645*10^-6$	$7.095*10^-6$
LSTM	$2.377*10^-5$	$1.043*10^-5$

6.4 Discussion of Model Performance

R^2 Was not presented in the above summary tables as it is not a reliable metric in non-linear model types. This is driven by the fact that sums of square errors do not behave the same in nonlinear models as they do in linear situations. As such only the mean squared error is presented as a metric for comparison [6].

In the case of all three pollutants of interest the autoregressive and LSTM models generalized much better outside of their training data than the regression models did. For the PM2.5 data the LSTM model was able to hit the lowest training mean squared error, although the autoregressive model did best in the test set. This might indicate that with some modification of the LSTM model it may be possible to better capture the behavior of PM2.5 than the other models. However, as is the behavior of LSTM PM2.5 model is unsatisfactory. As presented in the predictive modeling section, the behavior of the predicted graph significantly diverges from the actual time series ways that the autoregressive model does not. In all cases both by qualitative examination of the graphs (graphs of predictive model performance are including in the predictive modeling section) and by examination of the mean squared error against the test set the autoregressive model is the top performing model.

The autoregressive models perform well and even capture the seasonal trends despite not having incorporated a seasonal element (the relevant information was captured in the exogenous meteorological variables). However, particularly in PM2.5, there are some important discrepancies in the performance of the model relative to the actual data. On days when the value spiked particularly sharply, the model although also spiking significantly underestimated the value (a particularly good day to inspect for this behavior would be July 4, 2020 in the above graphs). This has significant implications on the value of the model in practical usage. Particularly, forecasting of poor air quality days is a key desired feature of air quality models and one in which the model is observed to consistently underestimate. As discussed in limitations, these spikes in PM2.5

are often caused by external factors not captured in meteorological data and as such are primarily captured by autoregression or the ability of an LSTM to discern time dependent changes.

The time series and LSTM models presented are forecasting 1 step out in a 24 hour window. As such, the performance of the models under longer forecast periods is not evaluated at this time. The ARX model could easily be modified to forecast out further in the future, but the new forecast times would require validation and would come at decreasing accuracy the further out the forecast window.

It is of note that the basic regression models do show modest predictive power despite using only weather data and not including a temporal aspect. This demonstrates that weather does have significant predictive value, and for the nitrogen dioxide concentrations was comparable in predictive power to the time series methods.

7 Limitations

Only meteorological data and air quality data on a daily basis were used for modeling. As such variables outside of this scope were not considered. The exploratory data analysis of the PM2.5 time series, revealed that factors such as wildfires and firework usage were highly significant factors in air quality; a more accurate model would require data on the fireworks usage and wildfire events. Furthermore, the data used is local to the area the forecast is being made, as wind speed was a significant factor a model containing surrounding locations and direction as well as speed would likely add significant predictive value. As discussed in the prior section, the time series models presented have only been trained and validating forecasting out 24 hours in the future; a practical use case for such models would likely require the ability forecast out several days which has not been evaluated in this study.

Hyperparameter tuning was only minimally explored due to time constraints and could likely lead to substantially better performance of several model types. Particularly, the structure of the LSTM model was only minimally varied and exploration of the number of LSTM nodes, hidden layers, and such were not fully undertaken. For the autoregressive models only a linear regression was studied, it is possibly that selecting a random forest regressor or other non-linear regressor might be able to improve performance.

8 Conclusion

The OpenAQ data set and NWS data were successfully extracted, transformed, and loaded into a local data warehouse. Exploratory analysis of the data suggested significant correlations between meteorological data and PM2.5, ozone, and nitrogen dioxide concentrations. Using these data sources regression models, autoregressive models, and LSTM models of modest predictive capacity were successfully developed for a chosen location within the data set. The models with

an autoregressive or temporal aspect were built on data on a twenty-four hour window predicting one step out. The models deliver reasonable error under this window and capture the qualitative aspects of the data sets, despite a tendency to underestimate extremely high particulate matter concentrations. This provides a case for the utility of these open data sets and modeling approaches for localized air quality forecasting. Of these models the autoregressive models performed the most accurately. All models struggled with handling sudden increases in particulate matter far above the normal, but accurately predicted the subsequent return to normal levels.

Future work might include both further tuning of the models as well as generalizing their deployment to the many locations in the data set. For the purpose of this project modeling and exploratory data analysis were focused on a well represented location in the data set, however, it is intended that this approach could serve as a prototype for the deployment of a system to model any given location within the data set. It is also an open question how well the developed models might generalize to different locations. If reliable data on fireworks usage, forest fires, and industrial emissions were included it is likely that the performance of the model could be refined further and would be a natural target for future data sources to integrate.

References

1. https://openaq.org/
2. https://registry.opendata.aws/openaq/
3. Bobcat fire scorches southern california, https://earthobservatory.nasa.gov/images/147324/bobcat-fire-scorches-southern-california
4. National centers for environmental information global surface summary of day data (gsod) (over 9000 worldwide stations), https://www.ncei.noaa.gov/data/global-summary-of-the-day/doc/readme.txt
5. Noaa global surface summary of day, https://registry.opendata.aws/noaa-gsod/
6. Why is there no r-squared for nonlinear regression?, https://blog.minitab.com/en/adventures-in-statistics-2/why-is-there-no-r-squared-for-nonlinear-regression
7. Aws-Samples: Aws-smsl-predict-airquality-via-weather, https://github.com/aws-samples/aws-smsl-predict-airquality-via-weather/blob/main/aq_by_weather.ipynb
8. Cordova, C.H., Portocarrero, M.N., Salas, R., Torres, R., Rodrigues, P.C., López-Gonzales, J.L.: Air quality assessment and pollution forecasting using artificial neural networks in metropolitan lima-peru. Scientific Reports **11**(1) (2021). https://doi.org/10.1038/s41598-021-03650-9
9. Fan, F., Lei, Y., Li, L.: Health damage assessment of particulate matter pollution in jing-jin-ji region of china. Environmental Science Pollution Research **26**(8), 7883 – 7895 (2019)
10. JoaquinAmatRodrigo: Joaquinamatrodrigo/skforecast: Time series forecasting with scikit-learn models, https://github.com/JoaquinAmatRodrigo/skforecast
11. Mapado: Mapado/haversine: Calculate the distance bewteen 2 points on earth, https://github.com/mapado/haversine

12. Padilla, J.: An introduction to data collection: Pulling openaq data from s3 using aws athena. (Jun 2020), https://johnathan-d-padilla.medium.com/an-introduction-to-data-collection-pulling-openaq-data-from-s3-using-aws-athena-26863b97c5cb
13. Petrowski, K., Bührer, S., Strauß, B., Decker, O., Brähler, E.: Examining air pollution (pm10), mental health and well-being in a representative german sample. Scientific Reports **11**(1), 1 – 9 (2021)
14. Santiago, J., Rivas, E., Gamarra, A., Vivanco, M., Buccolieri, R., Martilli, A., Lechón, Y., Martín, F.: Estimates of population exposure to atmospheric pollution and health-related externalities in a real city: The impact of spatial resolution on the accuracy of results. Science of The Total Environment **819**, 152062 (2022). https://doi.org/https://doi.org/10.1016/j.scitotenv.2021.152062, https://www.sciencedirect.com/science/article/pii/S0048969721071382
15. Schönleber, D.: Avoiding mistakes when working with json data in pandas (Jun 2021), https://dschoenleber.github.io/pandas-json-performance/