香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

MSBD 5002 - Knowledge Discovery and Data Mining

"Forecasting $PM_{2.5}$, PM_{10} and O_3 Pollution in Beijing, China"

Final Project

Group 27:

1. ZHENG Dongjia (ID: 20546139)
2. CHAN, Kai Leung Ken (ID: 20491619)

1. Introduction

Beijing - the capital, the political, the cultural center of China, and the key transportation hub of the world - has for many years suffered from severe short-term pollution events that are proven to be extremely harmful to human health. The dominant pollutants, particularly PM2.5, PM10 and O3 in haze pollution, are epidemiologically associated with the risk of deleterious health effects on cardiovascular and lung diseases. The adverse effects of these pollutants are well-documented, elevating public awareness to unprecedented levels and prompting intensive research targeting at simulating and forecasting its behavior so that proper measures can be taken to tackle this rising public health concern.

In this project, we attempt to leverage the commonly-used machine learning methods including eXtreme Gradient Boosting, (XGBoost) and Long Short Term Memory (LSTM) - Recurrent Neural Networks to establish the respective relationships between the concentration levels of PM2.5, PM10, O3 in the air quality data and the various selected meteorological(weather) data. Each method is evaluated against the common test dataset via Symmetric Mean Absolute Percentage Error (SMAPE) and the best-performing method is used to predict the concentration levels of PM2.5, PM10, O3, between May 1 to May 2, 2018 (over the coming 24*2) for the 35 stations in Beijing, China.

2. Data

2.1. Data Description

There are mainly two types of data in the project. The first type of data is the air quality data for Beijing from January 2017 to April 30, 2018. The second type of data is the meteorological (weather) data for Beijing from January 2017 to May 2, 2018.

For the air quality data, it consists of the hourly reading of the concentration level of PM2.5 (ug/m3), PM10 (ug/m3), NO2 (ug/m3), CO (mg/m3), O3 (ug/m3) and SO2 (ug/m3) from the 35 Beijing air quality stations. The following list shows the related files.

File name	Description
airQuality_201701-201801.csv	From January 2017 to January 2018
airQuality_201802-201803.csv	From February 2018 to March 2018
airQuality_201804.csv	April 2018
Beijing_AirQuality_Stations_en.xlsx	Location of air quality station in Beijing

As for the meteorological (weather Data), it consists of two types of data: Observed Weather Data and Grid Weather Data. They include the 651 points of grid weather data in Beijing and the observed weather data from 18 weather stations in Beijing.

File name	Description
observedWeather_201701-201801.csv	From January 2017 to January 2018
observedWeather_201802-201803.csv	From February 2018 to March 2018
observedWeather_201804.csv	April 2018
observedWeather_20180501-20180502.csv	observedWeather_20180501-20180502.csv
gridWeather_201701-201803.csv	From January 2017 to March 2018
gridWeather_201804.csv	April 2018
gridWeather_20180501-20180502.csv	May 1, 2018 to May 2, 2018(for prediction)
Beijing_grid_weather_station.csv	Location of grid weather station in Beijing
weatherData_detail.docx	Detailed information about weather data

2.2 Data Concatenation

Since there are a variety of tables and data types, it is essential to preprocess and join the tables into a simple format. The following list show the logical steps of this part: (noted that there are mainly 3 types of data tables: Air Quality data, Observed Weather data and Grid Weather data)

1) Firstly, we will simply analyze all the tables according to each type of the data so as to find out the missing time period for each data type.
2) Secondly, we will concatenate the tables according to each type of the data and finally get **3** simple tables: the Air Quality data table, Grid Weather data table and Observed Weather data table.
3) Thirdly, we will join the tables according to each air quality station, which means we will get **35** data tables for each air quality station.

2.2.1 Analyze the missing time period for each data type:

1) Air Quality data type:

File name	Start time	End time
airQuality_201701-201801.csv	2017-01-01 14:00:00	2018-01-31 15:00:00
airQuality_201802-201803.csv	2018-01-31 16:00:00	2018-03-31 15:00:00
airQuality_201804.csv	2018-04-01 02:00:00	2018-04-30 23:00:00

Conclusion: For air quality stations data, there is a time gap (11 hours) between 2018-03-31 15:00:00 and 2018-04-01 02:00:00.

2) Grid Weather data type:

File name	Start time	End time
gridWeather_201701-201803.csv	2017-01-01 00:00:00	2018-03-27 05:00:00

gridWeather_201804.csv	2018-04-01 00:00:00	2018-04-30 23:00:00
gridWeather_20180501-20180502.csv	2018-05-01 00:00:00	2018-05-02 23:00:00

Conclusion: For grid weather stations data, there is a time gap (about 5 days) between 2018-03-27 05:00:00 and 2018-04-01 00:00:00.

3) Observed Weather data type:

File name	Start time	End Time
observedWeather_201701-201801.csv	2017-01-30 16:00:00	2018-01-31 15:00:00
observedWeather_201802-201803.csv	2018-01-31 16:00:00	2018-04-01 00:00:00
observedWeather_201804.csv	2018-04-01 01:00:00	2018-04-30 23:00:00
observedWeather_20180501-20180502.csv	2018-05-01 00:00:00	2018-05-02 23:00:00

Conclusion: For observed weather stations, there is no time gap.

2.2.2 Concatenate the tables according to each data type

As for each data type, the tables are still separated into different time period. We have to concatenate the tables into a simple table with a union time period and finally get the three tables: Air Quality data table, Grid Weather data table, Observed Weather data table. **Noted that there may be some duplicate rows in the table and it is important to remove them before concatenation.**

a) Air Quality data table

After the concatenation of the air quality data table, we can get a simple table with time period between 2017-01-01 14:00:00 and 2018-04-30 23:00:00. There are 8 columns and 377,265 rows in this table. The following picture shows the content and the shape of the result:

	station_id	time	PM2.5	PM10	NO2	CO	O3	SO2
0	aotizhongxin_aq	2017-01-01 14:00:00	453.0	467.0	156.0	7.2	3.0	9.0
377264	dongsihuan_aq	2018-04-30 23:00:00	53.0	173.0	78.0	0.5	2.0	5.0

```
(377265, 8)
```

b) Grid Weather data table

After the concatenation of the grid weather data table, we can get a simple table with time period between 2017-01-01 00:00:00 and 2018-05-02 23:00:00. There are 8 columns and 7,529,400 rows in this table.

	station_id	temperature	pressure	humidity	wind_direction	wind_speed	time	weather
0	beijing_grid_000	-5.47	984.730	76.6	53.71	3.53	2017-01-01 00:00:00	None
7529399	beijing_grid_650	7.00	956.955	46.0	349.96	9.94	2018-05-02 23:00:00	CLEAR_DAY

```
(7529400, 8)
```

c) Observed Weather data table

After the concatenation of the observed weather data table, we can get a simple table with time period between 2017-01-30 16:00:00 and 2018-05-02 23:00:00. There are 8 columns and 197,051 rows in this table.

	station_id	time	temperature	pressure	humidity	wind_direction	wind_speed	weather
0	shunyi_meo	2017-01-30 16:00:00	-1.7	1028.7	15.0	215.0	1.6	Sunny/clear
197050	xiayunling_meo	2018-05-02 23:00:00	6.4	971.5	74.0	135.0	1.5	Sunny/clear

```
(197051, 8)
```

2.2.3 Concatenate the weather features from Grid Weather and Observed Weather data tables

In this project, weather features are of importance for the air quality prediction. Grid weather stations and observed weather stations provide the same kind of weather features. We can concatenate the weather features both these two kinds of the stations' data so that we can treat them together for further data preprocessing. The following picture shows the shape and content of the concatenated **Weather Data Table**.

	station_id	temperature	pressure	humidity	wind_direction	wind_speed	time	weather
0	beijing_grid_000	-5.47	984.73	76.6	53.71	3.53	2017-01-01 00:00:00	None
7726450	xiayunling_meo	6.40	971.50	74.0	135.00	1.50	2018-05-02 23:00:00	Sunny/clear

```
(7726451, 8)
```

2.3 Data Preprocessing

Data preprocessing plays a vital role in the whole project and it has a potential effect on the final prediction result. The purpose of this part is to transform the raw data into suitable format so that we can split the training dataset and testing dataset in the later steps. In order to predict the air quality for the 35 stations, we will use the weather data from K Nearest Weather Stations, including both the Grid Weather Stations and Observed Weather Stations, as the training features. The following steps show the procedures of features extraction.

2.3.1 Outliers

Before extracting the weather features, we have to cope with the outliers. There are several kinds of outliers in the dataset:

- Duplicated rows: simply use the function in Pandas named: '**drop_duplicates()**'
- The wind direction and wind speed with extremely large value (e.g. 999999 and 999017)
  a) In the project document, it is noted that if the wind speed is less than 0.5m/s (nearly no wind), the value of the wind direction is 999017. Thus, we will simply change these values into 0.5.

2.3.2 Features Extraction

In this part, we will calculate the K-Nearest Neighbors Weather Stations (including both Grid Weather Stations and Observed Weather Station) and extract the weather features for each air quality station. We will firstly try a K value of 3.

2.3.2.1 Get the location information for each station

1) From the following tables, we get the location for each air quality station, grid weather station and observed weather station
2) Join the location tables of grid weather stations and observed weather stations so as to get the **Weather Stations Location Table**.

Files name	Description
Beijing_grid_weather_station.csv	Location of grid weather station in Beijing
Beijing_AirQuality_Stations_en.xlsx	Location of air quality station in Beijing
observedWeather_201701-201801.csv	Contains the location for observed weather stations

2.3.2.2 Apply the Haversine Formula[1] to calculate the distance between stations.

$$\operatorname{hav}(\Theta) = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)$$

2.3.2.3 Import the Sci-Kit Learn Nearest Neighbors[2] to get the K Nearest Weather Stations

```
from sklearn.neighbors import NearestNeighbors
```

```
nn_finder = NearestNeighbors(n_neighbors=k_nn, metric='haversine', n_jobs=-1)
nn_finder.fit(stations_all_df)
result = nn_finder.kneighbors([[latitude, longitude]], return_distance=True)
```

The above code shows the main usage of this function. After the calculation we can get a **KNN-Stations-Table** to check up the K Nearest Neighbors for each air quality station. The following table shows the 2 nearest weather stations for 'dongsi_aq' and 'tiantan_aq' air quality station.

station_id	longitude	latitude	N0	N0_lat	N0_lng	N0_dist	N1	N1_lat	N1_lng	N1_dist
dongsi_aq	116.417	39.929	beijing_grid_303	39.9	116.4	195.880533	chaoyang_meo	39.9525	116.500833	364.123006
tiantan_aq	116.407	39.886	beijing_grid_303	39.9	116.4	92.912692	beijing_grid_324	39.9000	116.500000	356.975552

2.3.2.4 Weather features extraction

Since we have the **KNN-Stations-Table (in Step 2.3.2.3)** and the **Weather Data Table** (refers to the **Step 2.2.3)**, we can then extract the weather features for each air quality station from the K Nearest Weather Stations. The following picture shows the concatenated table for the air quality station named 'aotizhongxin_aq'. The air quality columns such as 'PM2.5', 'PM10', 'O3' can be regarded as the labels while the other weather data columns can be regarded as features for further model training and testing.

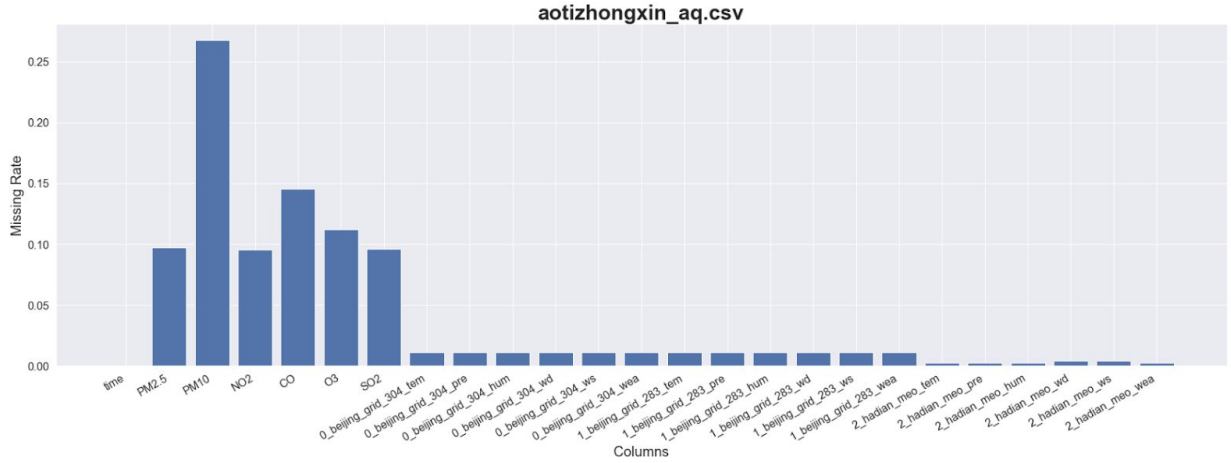
	time	PM2.5	PM10	NO2	CO	O3	SO2	0_beijing_grid_304_tem	0_beijing_grid_304_pre	0_beijing_grid_304_hum	...
0	2017-01-30 16:00:00	70.0	75.0	36.0	0.9	79.0	34.0	-5.89	1026.03	14.58	...
1	2017-01-30 17:00:00	78.0	86.0	36.0	0.1	78.0	38.0	-6.16	1025.68	15.11	...
2	2017-01-30 18:00:00	86.0	92.0	39.0	1.1	74.0	35.0	-6.44	1025.32	15.64	...

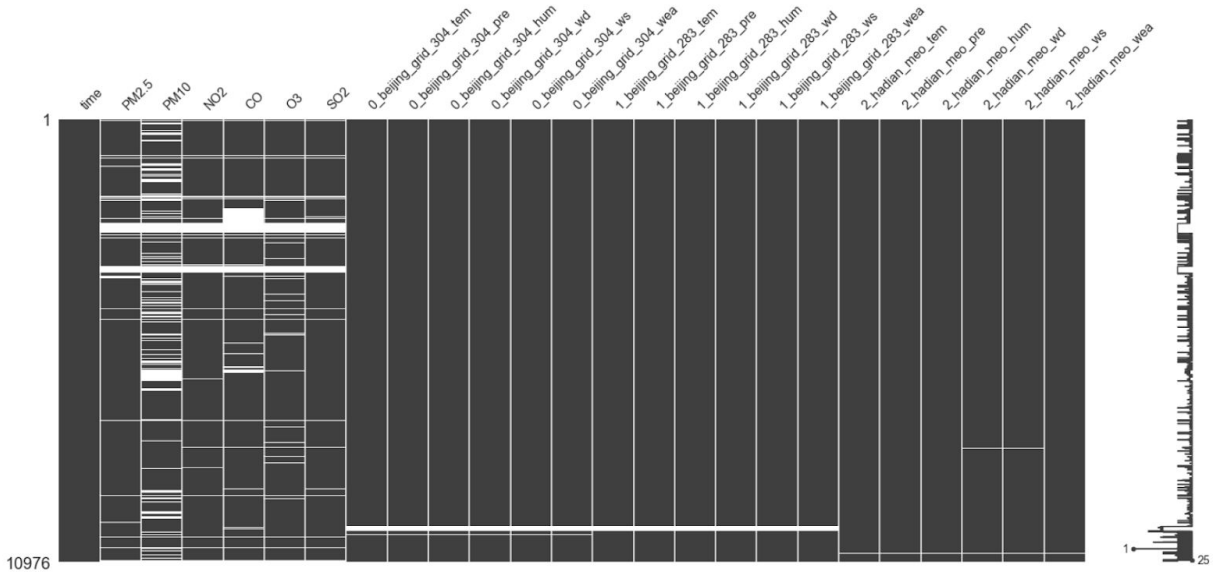
2.4 Data Exploration and Imputation

Right now, we have the preprocessed data table for each air quality station. From this step, we will make a thorough analysis for the preprocessed data table and try to fill the missing values. Data imputation is indispensable in this project because some of the air quality stations have a serious problem of data missing.

2.4.1 Missing rate comparison between Air Quality Labels and Weather Data Features

The following bar chart and matrix show the data missing rate and condition for each data column in the preprocessed table from **'aotizhongxin_aq'**. We choose **'aotizhongxin_aq'** because it is a typical one. For most of the air quality stations, their weather data missing rate is not high which is shown in the right part of the following bar chart and missing matrix. However, the air quality data such as 'PM2.5', 'NO2', 'O3', especially 'PM10' have a very high missing rate around 0.1 to 0.3, as shown in the left part in the following graphs.

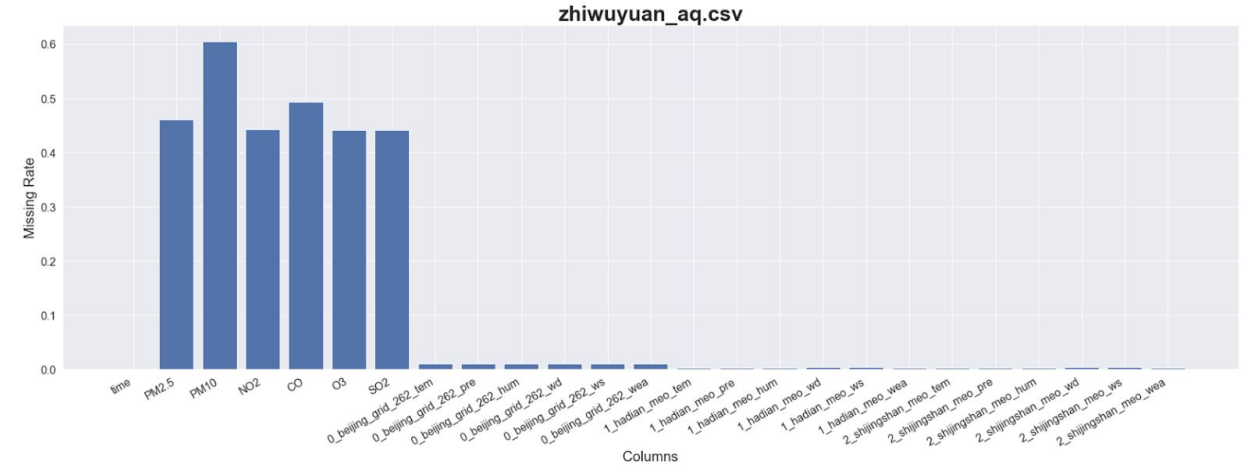


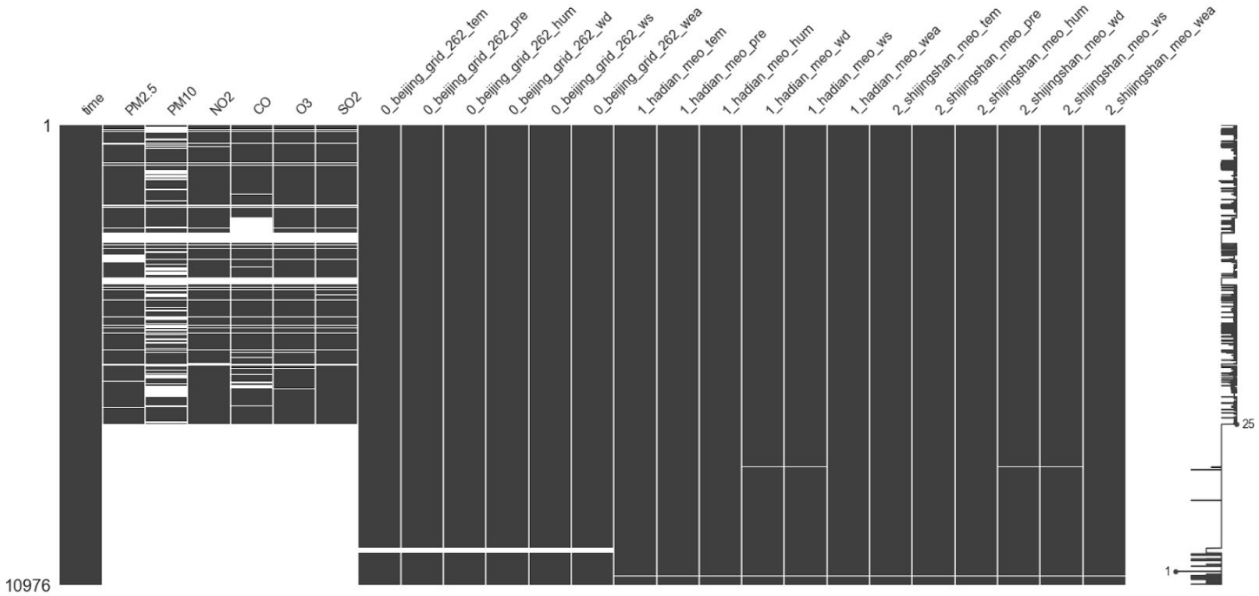


2.4.2 Missing rate comparison between Air Quality Stations

For most of the air quality stations, the air quality labels missing rate is around 0.1 and 0.3. However, for some of the air quality stations such as 'zhiwuyuan_aq', 'liulihe_aq', 'dongsihuan_aq', 'nansanhuan_aq', 'tongzhou_aq', the missing rates of their air quality labels

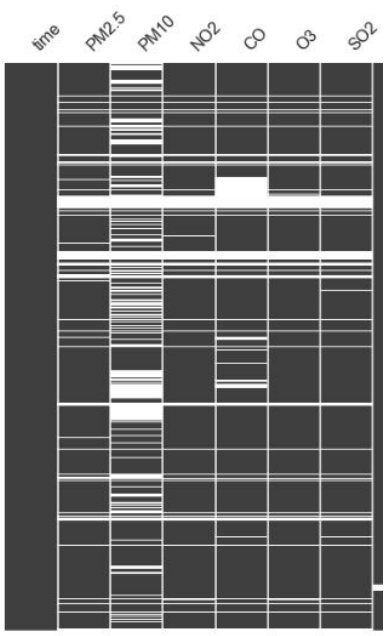
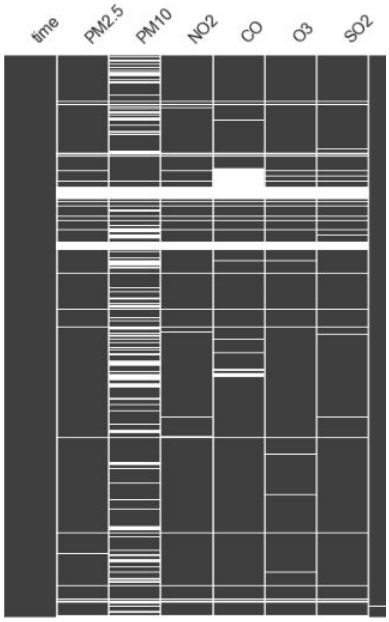
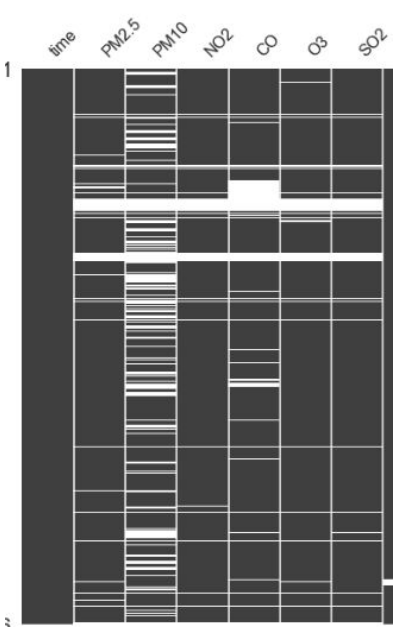
are extremely high, which is around 0.35 and 0.6. What's more, there is a big missing data gap which is astounding! 'zhiwuyuan_aq' is the typical one so we will use the following missing rate bar chart and matrix from 'zhiwuyuan_aq' for analysis.





2.4.3 Missing rate comparison between different Air Quality Types

For most of the air quality stations, the average missing rate of 'PM10' is very high compared with the other air quality type. The following table shows the detail missing rate situation for the randomly chosen stations 'yufa_aq', 'shunyi_aq' and 'yizhuang_aq'. It is obvious that there are a lot of blanks in the column of PM10.

yufa_aq	shunyi_aq	yizhuang_aq
		

2.4.4 Data imputation Strategy

From the analysis above, we can have an overview of the data missing condition of the dataset. But how to cope with the serious problems? We will follow the following strategies:

1) Use the KNN Data Imputation Algorithm[3] to fill the missing data concerning weather features.
2) Ignore the missing data concerning air quality labels such as 'PM10', 'PM2.5', 'O3' because their missing rate is too high and data imputation will cause bias for prediction.

3. Methods

In our project, we will mainly make a comparison between the models of **LSTM**(Long Short-Term Memory) and **XGBoost** (eXtreme Gradient Boosting). LSTM is a very popular RNN machine learning method in most of the weather and air quality prediction system because it can take the time sequence (e.g. 1 hour before, 1 day before or even 1 week before) into account and find out the variation tendency of the weather or air quality over time. As for XGBoost, it is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable.

Furthermore, in order to get the weather features from a specific time sequence, we have shifted the table so as to apply the method of **Sequence to Scaler** to all of the models. The following table gives an example of the final table for training and testing. 'var#' is the column name of the column variable and 't-#' shows the time index (e.g. 't-1' means 1 hour ago). The last column is the label column which represents the air quality such as 'PM2.5', 'PM10', 'O3' in this project.

	var1(t-1)	var2(t-1)	var3(t-1)	var4(t-1)	var5(t-1)	var6(t-1)	\
1	0.129779	0.352941	0.245902	0.527273	0.666667	0.002290	
2	0.148893	0.367647	0.245902	0.527273	0.666667	0.003811	
3	0.159960	0.426471	0.229508	0.545454	0.666667	0.005332	
4	0.182093	0.485294	0.229508	0.563637	0.666667	0.008391	
5	0.138833	0.485294	0.229508	0.563637	0.666667	0.009912	

	var7(t-1)	var8(t-1)	var1(t)
1	0.000000	0.0	0.148893
2	0.000000	0.0	0.159960
3	0.000000	0.0	0.182093
4	0.037037	0.0	0.138833
5	0.074074	0.0	0.109658

3.1 eXtreme Gradient Boosting (XGBoost)

The eXtreme Gradient Boosting (XGBoost) model is an implementation of the gradient boosting framework. Gradient Boosting algorithm is a machine learning technique used for building predictive tree-based models. Boosting is an ensemble technique in which new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. The ensemble technique uses the tree ensemble model which is a set of classification and regression trees (CART). The ensemble approach is used because a single CART, usually, does not have a strong predictive power. By using a set of CART (i.e. a tree ensemble model) a sum of the predictions of multiple trees is considered. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction.

The objective of the XGBoost model is given as:

$$\mathbf{Obj = L + \Omega}$$

Where,

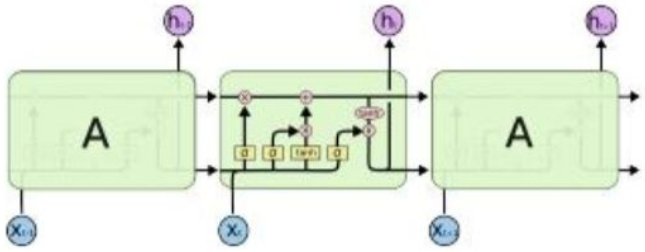
L is the loss function which controls the predictive power, and

Ω is regularization component which controls simplicity and overfitting

The loss function (L) which needs to be optimized can be Root Mean Squared Error for regression, Log loss for binary classification, or mlogloss for multi-class classification.

3.3 LSTM Sequence to Scalar

A long short-term memory (LSTM) Sequence to Scalar is built to analyze time series pollution data and predict the concentration level of PM2.5, PM10, O3, between May 1 to May 2, 2018 (over the coming 24*2) for the 35 stations in Beijing, China. LSTMs take as inputs not only the current input, but also what they have ”perceived” previously in time, essentially using the output at time t − 1 as an input to time t, along with the new input at time t.



Given this, the network effectively has ’memory,’ unlike feedforward networks. This characteristic is important because there is often information in the sequence itself, and not just the outputs. With the time series pollution and meteorological data from 2015 to 2017 LSTM Sequence to Scalar is an appropriate model for air quality prediction. In this method, we also altered the number of previous hours (timesteps) used for the previous sequence, using 3, 5 , 8 and 10 hours (timesteps). We also experimented various network configurations, including number of layers, number of nodes per layer, loss functions, batch size, and number of epochs, to improve the accuracy.

4.0. Results

It is quite tough to make a comparison for each air quality station, so we simply choose the data from a typical air quality station name **‘aotizhongxin_aq’** for the models comparison and the following parameters tuning.

4.1 Evaluation Metrics

In our project, we mainly use the evaluation metrics of RMSE and SMAPE. The formula for each of the metric is listed below:

- **RMSE** (Root-Mean-Square Error)

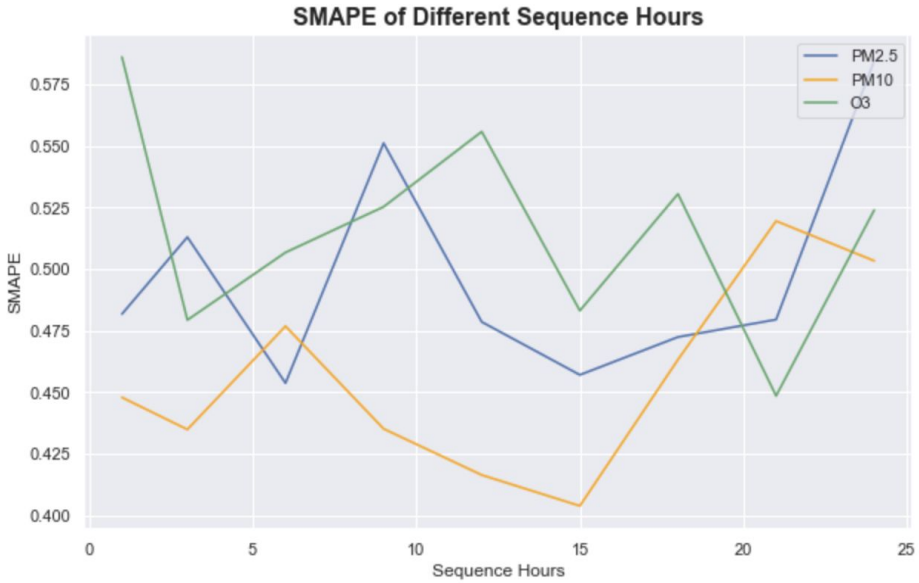
$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

- **SMAPE** (Symmetric Mean Absolute Percentage Error)

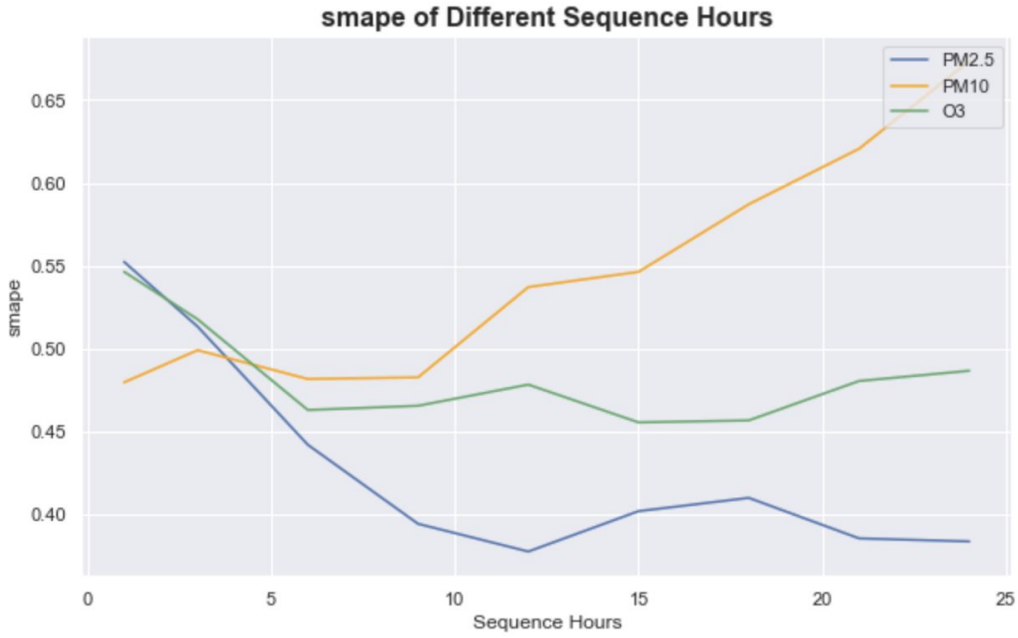
$$\mathrm{SMAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}$$

4.2 Models Comparison

4.1.1 LSTM

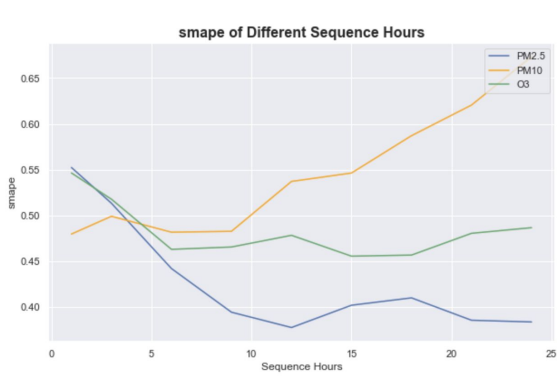


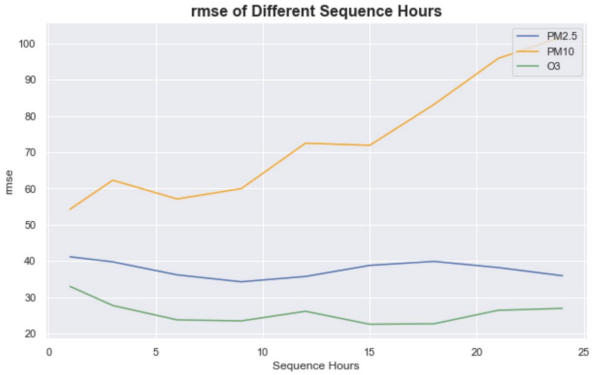
4.1.2 XGBoost



Conclusion: From the above graphs, it is obvious that XGBoost has a better performance and SMAPE result compared with LSTM. Thus, we will use the method of **XGBoost for** prediction.

4.3 Suitable Time Sequence for Prediction





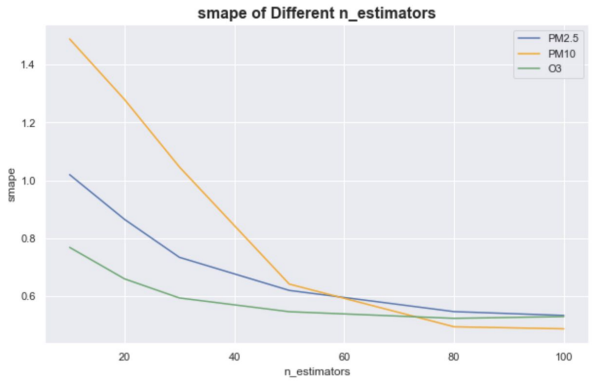
The above graphs is the SMAPE and RMSE score from the model of XGBoost. The blue, orange, green lines represents the score of PM2.5, PM10, O3, respectively.

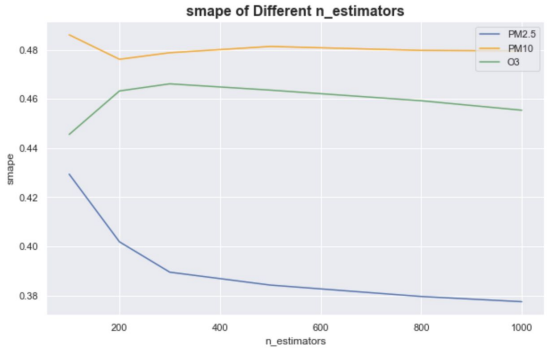
Conclusion:

- For **PM2.5(the blue line)**, the best sequence hours is around **12**.
- For **PM10(the orange line)**, the best sequence hours is around **1**.

- For **O3(the green line)**, the best sequence hours is around **15**.

4.4 Suitable n_estimators for XGBoost



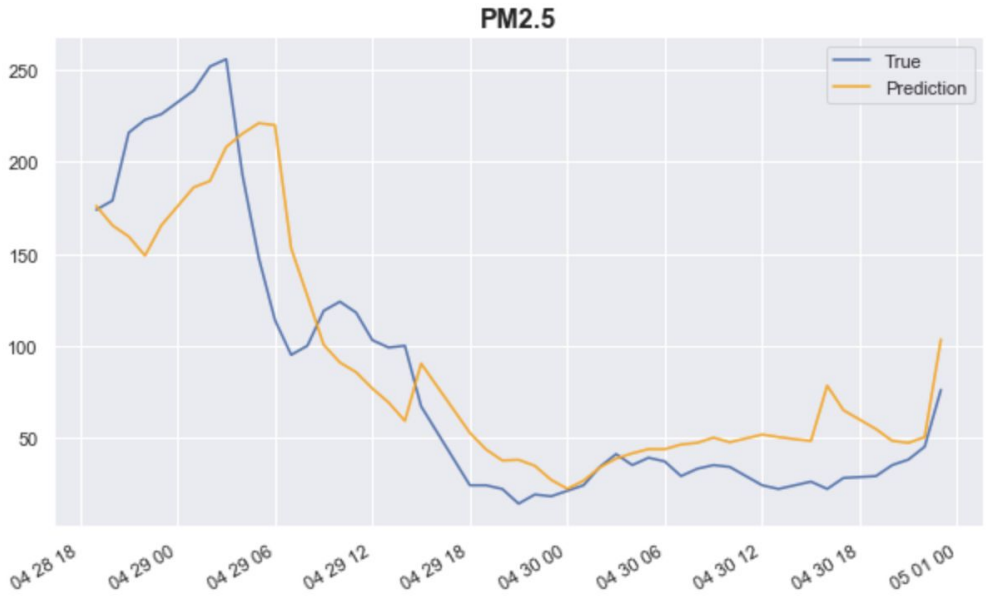


The graphs above shows the relationship between SMAPE and the parameter of 'n_estimators' for XGBoost. It seems for all the air quality types, the value of SMAPE decreases sharply from 10 estimators to 100 estimators and then the SMAPE starts to increase. To draw a conclusion:

- For **PM2.5(the blue line)**, the best n_estimators is around **1000**.
- For **PM10(the orange line)**, the best n_estimators is around **200**.
- For **O3(the green line)**, the best n_estimators is around **100**.

4.5 Test Result

Finally, we test on the data from station 'aotizhongxin_aq' between 2018,04,29 00:00 and 2018,04,29 00:00 (48 hours). The prediction line charts are shown in the below table. Noted that the blue line represents the true value while the orange line represents the prediction value.

PM2.5	This took 62.64255499839783 seconds. MSE: 1275.712363093039 RMSE: 35.71711582831177 SMAPE: 0.3774762397648348 R^2 score: 0.766573 
PM10	This took 1.7225351333618164 seconds. MSE: 2914.831589138897 RMSE: 53.98918029697151 SMAPE: 0.4761605575637878 R^2 score: 0.332538 

O3	This took 8.286745071411133 seconds. MSE: 394.6390065084467 RMSE: 19.865523061536706 SMAPE: 0.4455550562993526 R^2 score: 0.877812 

5.0. Conclusion

To draw a conclusion, the best model we have ever tried is the XGBoost. It gives the best SMAPE score compared with LSTM.

Meanwhile, the suitable time sequence hours for PM2.5, PM10 and O3 are different from each other. From the graph in **Section 4.1.2**, we can see that for PM2.5 and O3, the SMAPE value decreases as the time sequences is getting bigger. However, PM10 gives an opposite situation. Perhaps for the air quality of PM10 data, its values missing rate is very high (shown in **Section 2.4.3**) so when we are using the Sequence to Scaler method in the models, the error tends to be increasingly high as the time sequence is bigger. However, there is no such problem for PM2.5 and O3.

Last but not the least, the overall testing SMAPE score for PM2.5 is the smallest while the overal testing SMAPE score for P10 is the highest among the three types of air quality, PM2.5, PM10 and O3. There should be a lot of advanced methods to improve the prediction accuracy for PM10. Thus, just stay hungry and stay foolish!

Reference

[1] Haversine Formula on Wikipedia

[2] Sklearn Nearest Neighbors

[3] Imputation Algorithms and Packages from Fancyimpute

[4] Support Vector Machine - Regression (SVR)

[5]Forecasting using eXtreme Gradient Boosting (XGBoost)

[6] XGBoost Documentation

[7] Multivariate Time Series Forecasting with LSTMs in Keras

[8] Prediction of air quality indicators for the Beijing-Tianjin-Hebei region, Lifeng Wu, NuLi,Yingjie Yang

[9] Deep Air: Forecasting Air Pollution in Beijing, China