MSDB5002 PROJECT REPORT

Group number: 12
YANG Yue   WANG Yutong
20549600   20541402

Abstract

The project is required to predict concentration levels of three kinds of pollutants over 48 hours for 35 stations in Beijing, and the given data information including the air quantity, and the grid and observed stations of weather conditions over a year in different places in Beijing. After cleaning and processing the data, the model can be established. Based on that, the pollution level of PM2.5, PM10, O3 for the two days can be predicted where need to be predicted once a time for 48 hours for each station, so finally there are 5040 predictions in total.

1. Introduction

The project is required to predict concentration levels of three kinds of pollutants over 48 hours for 35 stations in Beijing. There are several different kinds of data available and the given data information including the air quantity, and the grid and observed stations of weather conditions. The air quality data contains the concentration of PM2.5, PM10, NO2, CO, O3 and SO2 from Beijing and there are three files with different months period, besides, a Beijing_airquality_stations.xlsx shows the summary of the different locations of the air quality stations. On the other hand, the weather data contains 2 types of data. Firstly, the grid weather contains 651 points in Beijing according to the range of longitude and latitude, Secondly, the observed weather files show 18 different location of weather stations.

The weather data includes the weather condition, humidity, pressure, temperature, wind direction, wind speed and all the information can be regarded as features. Another station map shows the details of all the stations visually, including grid weather points, air quality stations and observed stations about their longitudes and latitudes. After cleaning, processing and analyzing the data, the model can be established. All the information is on hourly basis. Based on that, the pollution level of PM2.5, PM10, O3 for the two days can be predicted, and the results show that once an hour between May 1 to May 2,2018 for 35 air quality stations.

2. Data description

There are three types of data given in the project. First, there are 35 stations of the air quality data, and the air quality data contains the concentration of PM2.5, PM10, NO2, CO, O3 and SO2 from Beijing and there are three files with different months period, besides, a Beijing_airquality_stations.xlsx shows the summary of the different locations of the air quality stations. Secondly, the weather data contains 2 types of data. On the one hand, the grid weather contains 651 points in Beijing according to the range of longitudes and latitudes. On the other hand, the observed weather files show 18 different location of weather stations. The weather data includes the weather condition, humidity, pressure, temperature, wind direction, wind speed and both the grid and observed weather give data information among 15 months for 3 files each. All the weather information given is on hourly basis. Another station map shows the details of all the stations visually, including grid weather points, air quality stations and observed stations about their longitudes and latitudes.

3. Feature Engineering

Feature engineering is the process of using different ways of the data to create features that make machine learning algorithms work and feature engineering is fundamental to the application of machine learning. As a result, feature engineering is an essential part of building any intelligent system.

3.1. Data Preprocessing

The data processing is necessary due to the data given us are generally incomplete which lacking attribute values, lacking certain attributes, or containing only aggregate data.

Firstly, we can see the map below shows the different locations of stations, and the grey points indicate the grid weather points which is uniformly distributed due to the longitudes and latitudes. The air quantity stations which are used for predictions are shown with red points and the dark points represent the observer stations.

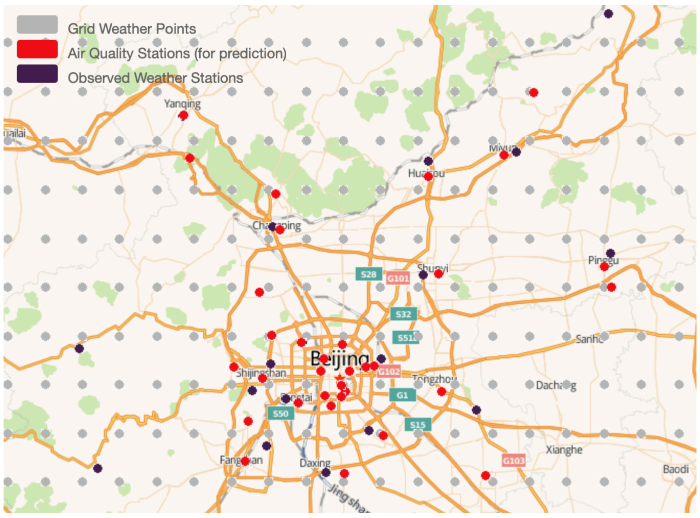


Figure 1. The locations of stations

In this case, the method we use is that choose the closest observed stations and grid points as references to predict the air quality stations for the reason that the pollution or the weather condition will not vary too much in a small area.

3.1.1 Weather data processing

The six files of the weather data including 3 grid weather files and 3 observed weather files need some data processing. The steps are as follows.

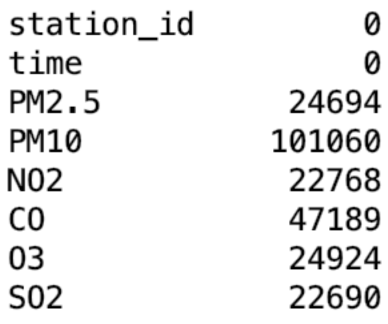
- The weather conditions in grid weather for 201701-201803 are missing, the solution is to fill the NaN values with the weather values of the close observed station at the same time period.

- In order to keep the data uniform, we drop the columns of longitude and latitude in the grid_weather in 201701-201803.
- The name of the columns should be the same, so change the columns' names to the same standard. For example, transfer the "utc_time" and "time" into "time"
- Combine all the weather data together with all the months.
- For the missing values of the "humidity", "pressure", "temperature", "wind_speed", and "wind_direction", we deal with using the mean value of the same station in the same day to fill in.
- Use the mode of the same day in the same station to fill in the missing data of the weather.
- Another file with the predicted days weather data is done with the same processing.

3.1.2. Air quality data processing

There are 3 files containing 15 months air quality for 35 stations in Beijing. The columns including the concentration of PM2.5, PM10, NO2, CO,O3 and SO2. However, some data are missing and need some processing. The steps are as followings.

- Generate the air quality data according the time sequence.
- Notice that for some stations, they miss several days or even months data, so remove the data where all day is NaN.
- There are many values missing in air quality files, and the picture below shows the number of the missing data.



```
station_id        0
time              0
PM2.5         24694
PM10         101060
NO2           22768
CO            47189
O3            24924
SO2           22690
```

Figure 2. Missing data in air_quality

- The picture below shows the relationship between the features which is the coefficient of association for each pollutant.

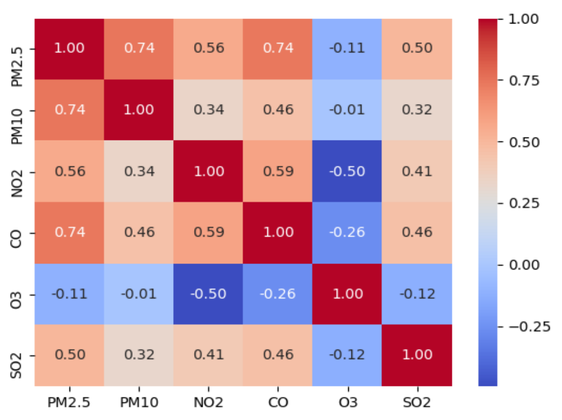
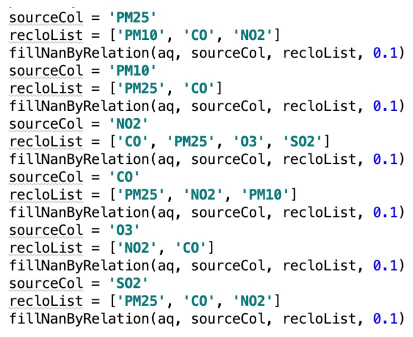


Figure 3. The relationship between features

As we can see, the PM2.5 and PM10 have strong connections, and each feature seems to have close connections with each other.

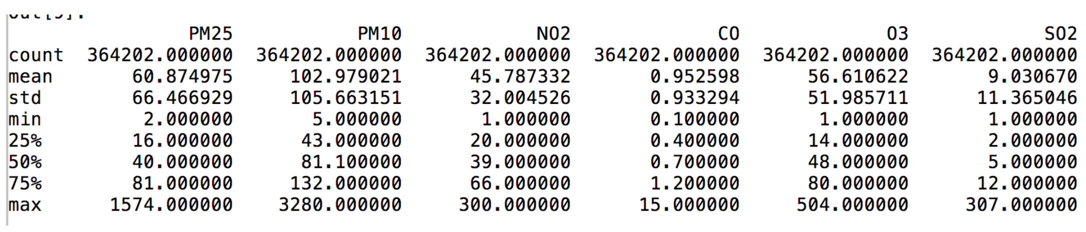
- Fill the NaN values based on the feature connections.



```
sourceCol = 'PM25'
recloList = ['PM10', 'CO', 'NO2']
fillNanByRelation(aq, sourceCol, recloList, 0.1)
sourceCol = 'PM10'
recloList = ['PM25', 'CO']
fillNanByRelation(aq, sourceCol, recloList, 0.1)
sourceCol = 'NO2'
recloList = ['CO', 'PM25', 'O3', 'SO2']
fillNanByRelation(aq, sourceCol, recloList, 0.1)
sourceCol = 'CO'
recloList = ['PM25', 'NO2', 'PM10']
fillNanByRelation(aq, sourceCol, recloList, 0.1)
sourceCol = 'O3'
recloList = ['NO2', 'CO']
fillNanByRelation(aq, sourceCol, recloList, 0.1)
sourceCol = 'SO2'
recloList = ['PM25', 'CO', 'NO2']
fillNanByRelation(aq, sourceCol, recloList, 0.1)
```

Figure 4. Fill the air_quality NaN values

In this case, we can get the air_quality data without missing values and the brief information about the air_quality is as follow.



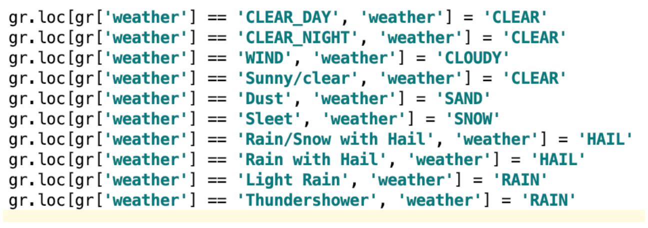
```
                PM25           PM10            NO2             CO             O3            SO2
count  364202.000000  364202.000000  364202.000000  364202.000000  364202.000000  364202.000000
mean       60.874975     102.979021      45.787332       0.952598      56.610622       9.030670
std        66.466929     105.663151      32.004526       0.933294      51.985711      11.365046
min         2.000000       5.000000       1.000000       0.100000       1.000000       1.000000
25%        16.000000      43.000000      20.000000       0.400000      14.000000       2.000000
50%        40.000000      81.100000      39.000000       0.700000      48.000000       5.000000
75%        81.000000     132.000000      66.000000       1.200000      80.000000      12.000000
max      1574.000000    3280.000000     300.000000      15.000000     504.000000     307.000000
```

Figure 5. The air_quality NaN description.

3.2. Feature Selection

Feature selection, also known as variable selection which is the process of selecting a subset of relevant features for use in model construction. After dealing with the data, the feature selection and processing are the following one of the most important step.

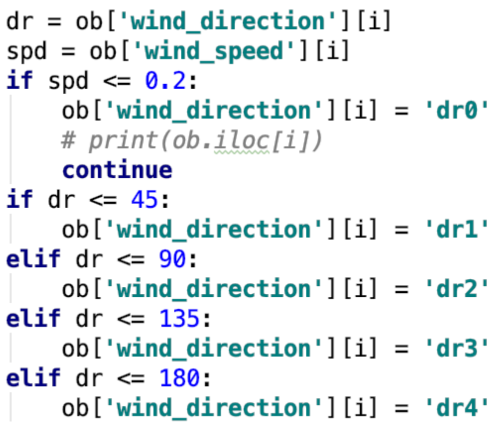
There are many different types of weather in each file, and the classes of weather different from each other. As a result, we need to classify and combine the close classes and choose some of them as features. The figure 6 shows some samples.



```
gr.loc[gr['weather'] == 'CLEAR_DAY', 'weather'] = 'CLEAR'
gr.loc[gr['weather'] == 'CLEAR_NIGHT', 'weather'] = 'CLEAR'
gr.loc[gr['weather'] == 'WIND', 'weather'] = 'CLOUDY'
gr.loc[gr['weather'] == 'Sunny/clear', 'weather'] = 'CLEAR'
gr.loc[gr['weather'] == 'Dust', 'weather'] = 'SAND'
gr.loc[gr['weather'] == 'Sleet', 'weather'] = 'SNOW'
gr.loc[gr['weather'] == 'Rain/Snow with Hail', 'weather'] = 'HAIL'
gr.loc[gr['weather'] == 'Rain with Hail', 'weather'] = 'HAIL'
gr.loc[gr['weather'] == 'Light Rain', 'weather'] = 'RAIN'
gr.loc[gr['weather'] == 'Thundershower', 'weather'] = 'RAIN'
```

Figure 6. Sample code for “weather” processing.

Another feature need to be processing is the wind direction. The wind direction given is a value which lies in 0 to 360, in order to make the wind directions more direct and easier to fit the model, and we transfer the wind direction into another form of features. The directions can be classified into 9 choices, the first range is 0-45 where the direction is d1 and increasing with 45 stage. Notice that when the wind speed is close to 0 which means there is no wind, the wind direction should be dr0. The sample code is as follows.



```
dr = ob['wind_direction'][i]
spd = ob['wind_speed'][i]
if spd <= 0.2:
    ob['wind_direction'][i] = 'dr0'
    # print(ob.iloc[i])
    continue
if dr <= 45:
    ob['wind_direction'][i] = 'dr1'
elif dr <= 90:
    ob['wind_direction'][i] = 'dr2'
elif dr <= 135:
    ob['wind_direction'][i] = 'dr3'
elif dr <= 180:
    ob['wind_direction'][i] = 'dr4'
```

Figure 7. Sample code for “wind_direction” processing

For the reason that we need to train 35 stations for 48 hours about three pollutants, there are so many features that cannot output at the same time. After analyzing and balancing the solutions, we use past time weather conditions as features and use several days to predict 48 hours for each station.

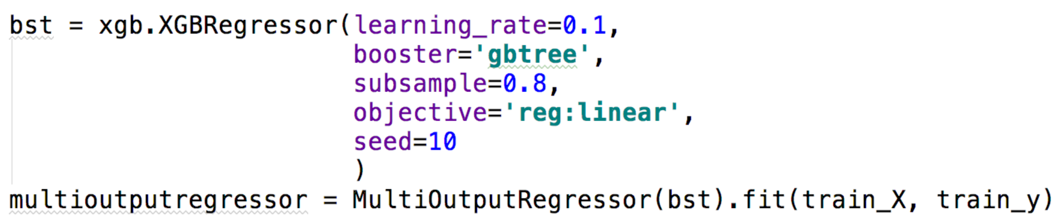
4. Models and Outcomes

4.1. Model Selection

The design of input and output is to input the weather data and air. pollution data of several days, and the model should predict the air pollution data for the next 2 days. Using the generation model sequence to sequence can solve the problem of variablelength sequence effectively. In addition, we also consider LSTM which is very suitable for time series prediction, and XGBoost with MultiOutputRegressor.

4.1.1. XGBoost with MultiOutputRegressor

XGBoost stands for e**X**treme **G**radient **B**oosting. It is an. implementation of gradient boosted decision trees designed for speed and performance. The two reasons to use XGBoost are also the two goals of the project: execution speed and model performance. Generally, XGBoost is really fast when compared to other implementations of gradient boosting. Although the XGBoost regression can only output single target, we can use the MultiOutputRegressor supported by Scikit-Learn to output multiple targets. MultiOutputRegressor consists of fitting one regressor per target. It is a simple strategy for extending regressors that do not natively support multi-target regression. The following figure 8 shows the code that implements a XGBoost regressor with MultiOutputRegressor.

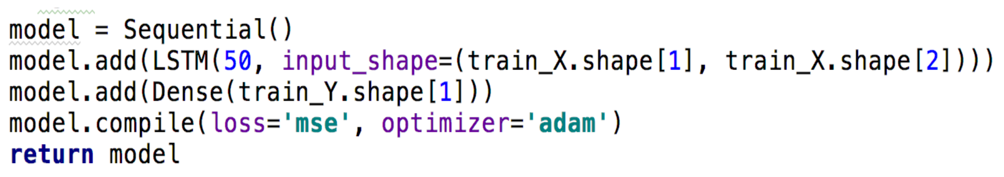


```
bst = xgb.XGBRegressor(learning_rate=0.1,
                       booster='gbtree',
                       subsample=0.8,
                       objective='reg:linear',
                       seed=10
                       )
multioutputregressor = MultiOutputRegressor(bst).fit(train_X, train_y)
```

Figure 8. XGBoost Structure Code

4.1.2. LSTM

LSTMs (Long-Short Term Memory) 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 [1], along with the new input at time t. Therefore, 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 [2]. Air pollution varies temporally, so the best predictor of future air pollution is previous air pollution over long time periods. Hence, LSTM is well suitable for predicting the air pollution. The Figure 9 shows the code that implements a simple LSTM structure.



```
model = Sequential()
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(train_Y.shape[1]))
model.compile(loss='mse', optimizer='adam')
return model
```

Figure 9. LSTM Structure Code

4.1.3. Sequence to Sequence

Similar to LSTMs, sequence to sequence is also a model structure derived from RNN (Recurrent Neural Network). A typical sequence to sequence model has two parts – an encoder and a decoder, shown as the Figure 10. Both the parts are practically two different neural network models combined into one giant network. Broadly, the task of an encoder network is to understand the input sequence, and create a smaller dimensional representation of it. This representation is then forwarded to a decoder network which generates a sequence of its own that represents the output.

In this project, we use Encoder-Decoder based on GRU(Gate Recurrent Unit). GRU is one of the RNN networks. Like LSTM, it is also proposed to solve problems of long-term memory and the gradients in back propagation. Compared to LSTM, using GRU can achieve more considerable results. It is easier to train, and it can greatly improve the training efficiency, hence finally we choose GRU. The Figure 11 shows the code that implements an Encoder-Decoder structure based on GRU.

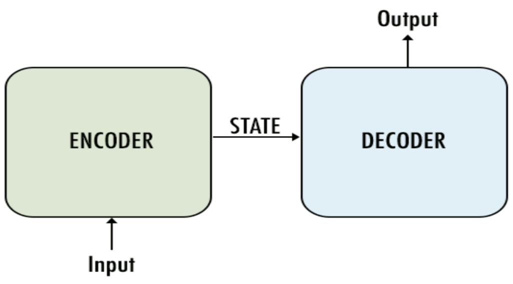
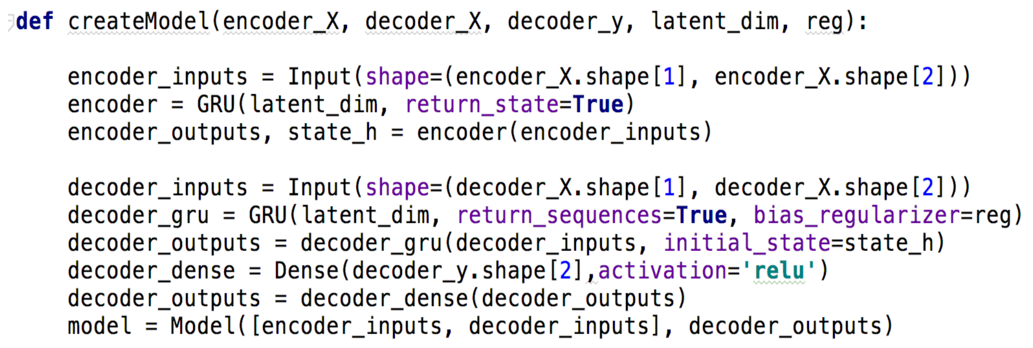


Figure 10. Encoder-Decoder structure



```
def createModel(encoder_X, decoder_X, decoder_y, latent_dim, reg):

    encoder_inputs = Input(shape=(encoder_X.shape[1], encoder_X.shape[2]))
    encoder = GRU(latent_dim, return_state=True)
    encoder_outputs, state_h = encoder(encoder_inputs)

    decoder_inputs = Input(shape=(decoder_X.shape[1], decoder_X.shape[2]))
    decoder_gru = GRU(latent_dim, return_sequences=True, bias_regularizer=reg)
    decoder_outputs = decoder_gru(decoder_inputs, initial_state=state_h)
    decoder_dense = Dense(decoder_y.shape[2],activation='relu')
    decoder_outputs = decoder_dense(decoder_outputs)
    model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
```

Figure 11. Sequence to Sequence based on GRU

4.1.4. Selection

In these three type of models, comparing the run time and validation errors, we finally chose a slightly better model sequence to sequence base on GRU. Training time comparison is LSTM < Seq2Seq < XGBoost, and the training performance level is Seq2Seq > XGBoost > LSTM. We had considered to integrate these models, but due to the limited time, we just selected single model to train.

4.2. Structure

Sequence to sequence is chosen in our project. We design to input the weather data and air pollution data of 4 days to the encoder, and the decoder take encoder state and the weather of next 2 days as input. Finally, the decoder output the air pollution data for the next 2 days. The detail structure information is shown as the Figure 12.

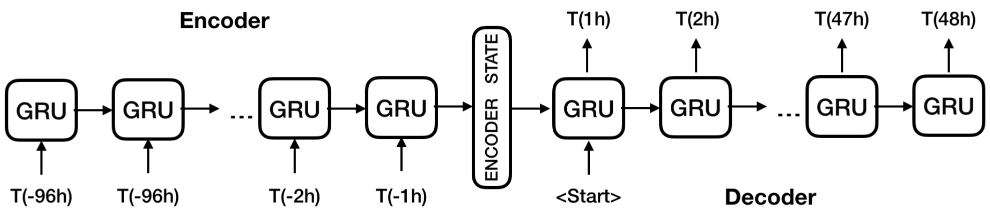
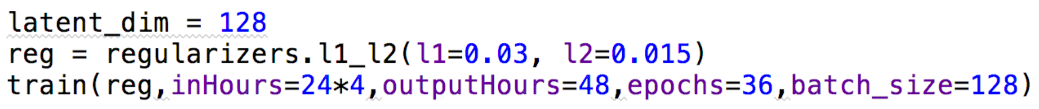


Figure 12. Model Structure

4.3. Training and Outcomes

We trained a model for each site, totally we trained 35 models. During the training, we found that it is quite easy to perform over-fitting. The training error is reduced to about 20, but the test error is still around 60, and there is a rising trend of test error in the later epochs. So, we added regularity to the GRU layer, and adjusted the parameters of the regular. Additionally, we adjusted parameters such as batch size, epochs, and latent_dim, regularization. The figure 13 shown the finalized parameters. EarlyStopping is also used in this project to prevent more inefficient epochs. The final result is shown as Figure 14.



```
latent_dim = 128
reg = regularizers.l1_l2(l1=0.03, l2=0.015)
train(reg,inHours=24*4,outputHours=48,epochs=36,batch_size=128)
```

Figure 13. Parameters

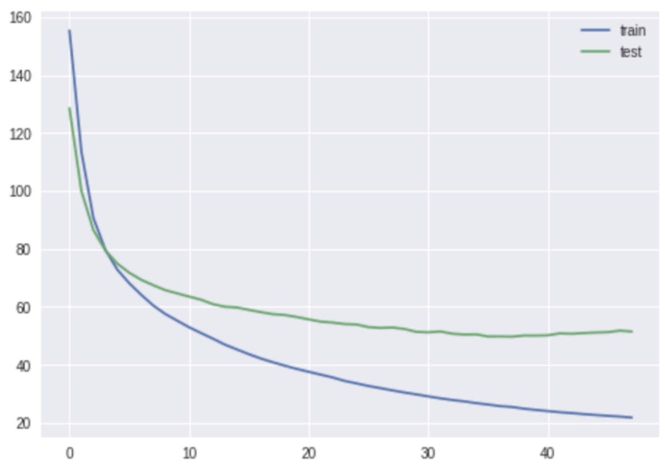


Figure 14. Results

5. Conclusion

In this project, the data source is quite complicated, covering both time and space dimension. The data has noise and instability. The pollution data to be predicted is weakly regular and it is easy to be abrupt. Long time series data is difficult for feature extraction and modeling.

In the data preprocessing, a lot of data clean, data integration and empty filling work has been done, for example, finding the grid weather sites and the observed sites near the air quality sites, filling the empty value of grid weather for one year, integrating weather data and so on. There are also many empty values in the air quality data, which are filled with other highly relevant contaminants.

In the model selection and parameter adjustment, we investigated some models and selected three, including XGBoost, LSTM, Sequence to Sequence. According to the results and run time, we chose Sequence to Sequence. The original intention was to train three type of models separately, and finally integrated these models,. But due to the limited time, we just trained one model. Based on the weather and air pollution data for the first 96 hours and the weather forecast for the next 48 hours, it is able to predict the air quality for the next 48 hours, we basically reach the target.

6. Future Work

There is still more room for improvement. For example, for the feature extraction of the spatial topology, we can consider more, such as taking the data of 8 grid sites near the air quality site (only two or three are selected in our project), and the air has fluidity, we can consider the data sampling of the larger span of the whole city.

In the empty value filling, such as grid filling weather, we also need to consider the noise and the different distribution between the forecast weather and the actual weather.

In the feature selection, considering the number of features is very large (2 thousand), decision trees can be used to select some most important features, and PCA can also be used to reduce dimensions. In model training, we can train several models and finally merged them. There are still a lot of things that can be done, and we will continue to optimize it, hoping that the predictions will be more accurate.

7. Task assignments

Name	Task Assignment
WANG Yutong	Data Preprocessing: Air quality data Model Implement and Training: LSTM
YANG Yue	Data Preprocessing : Observed Weather Data, Grid Weather Data Model Implement and Training: Seq2Seq, XGBoost

Reference

[1] Sepp Hochreiter and Jurgen Schmidhuber. “Long Short-Term Memory”. In: ¨ Neural Comput. 9.8 (Nov. 1997), pp. 1735– 1780. ISSN: 0899-7667.
[2] Abdelhadi Azzouni and Guy Pujolle. “A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction”.