MSBD5002 Group Project

Wang Chuhan(ID:20550752), Hui Kwat Kong(ID:20123133)

December 6, 2018

1 Introduction

1.1 Task Description

Predict concentration levels of several pollutants over the coming 24*2 hours (two days) for 35 stations in Beijing, China.

1.2 Data Description

Air Quality Data (on hourly basis) The Air Quality Data file contains the concentration of PM2.5 (ug/m3), PM10 (ug/m3), NO2 (ug/m3), CO (mg/m3), O3 (ug/m3) and SO2 (ug/m3) from Beijing.

Weather Data (on hourly basis) The Weather Data contains 2 types of data: Observed Weather Data and Grid Weather Data. 651 points of grid weather data in Beijing and observed weather data from 18 weather stations in Beijing are provided.

2 Data Preparation

2.1 Data Exploration

First of all, a preliminary analysis was performed for understanding the data. After loading the data as DataFrame, we applied head() method to check the top five rows and info() method for quick description of the data, in particular, the total number of rows, and each attribute type and the number of non-null values.

For illustration, Fig.1 and Fig.2 show the air quality from Jan 2017 to Jan 2018. There are total eight columns and six of them having missing values.

	stationId	utc_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
1	aotizhongxin_aq	2017-01-01 15:00:00	417.0	443.0	143.0	6.8	2.0	8.0
2	aotizhongxin_aq	2017-01-01 16:00:00	395.0	467.0	141.0	6.9	3.0	8.0
3	aotizhongxin_aq	2017-01-01 17:00:00	420.0	484.0	139.0	7.4	3.0	9.0
4	aotizhongxin_aq	2017-01-01 18:00:00	453.0	520.0	157.0	7.6	4.0	9.0

Figure 1

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 311010 entries, 0 to 311009
Data columns (total 8 columns):
stationId    311010 non-null object
utc_time     311010 non-null object
PM2.5        290621 non-null float64
PM10         227747 non-null float64
NO2          292359 non-null float64
CO           268197 non-null float64
O3           290589 non-null float64
SO2          292462 non-null float64
dtypes: float64(6), object(2)
```

Figure 2

2.2 Data Integration

After having a first glance at our data, we then combined the data from different resources to a unified view. We first combined the three Air Quality data frames of different time periods using pandas.concat() function. The same method has been applied to combine the grid weather data and observed weather data similarly.

Since there are missing in the Air Quality data, we first created a based dataframe having all timestamp from 1st Jan 2017 to 30th April 2018 for each station.

Finally we merge the air quality data, grid weather data and observed weather data to the based dataframe according to location(longitude, latitude) and utc_time using df.merge() function (left join).

3 Feature Generation

Before actually preparing the data for machine learning algorithm, it is critical to perform feature engineering based on domain knowledge and attribute combinations.

3.1 Direct timestamp information

One of our methodologies is to create new attributes based on time. We created attributes such as year, month, hour, weekday, holiday. Note that the holiday represents whether that day is a national holiday promulgated by the State Council.

3.2 Information from previous timestamp

Another strategy is to make use of the information in previous timestamp for each station. For example, air quality index in the previous hours should affect the air quality index in the next hour. Therefore, we generated new attributes based on the weather conditions and pollution measurements over the last several hours, such as PM2.5(t-1), PM2.5(t-2) and so

on. We defined a shifting() function to generate these features. Before we created the shifted columns, it is necessary to fill the missing values first:

3.2.1 missing data treatment

For numerical data, as air pollution measurements are time sequential in each station(we sorted the data by station and time), we applied interpolation method to replace the missing value (it replaces the missing value by the mean of nearest timestamp under each station). For categorical data, we applied ffill method (in pandas.DataFrame.fillna) to fill the missing value, which propagates the last valid observation forward to next. This is assumed that our data is time sequential continually, however, there are some time gap in the raw data. For example, the next record of autizhongxi station at 23:00pm on 1st Feb in 2017 is 12:00am on 1st March in 2017, with one month gap between these two records. Hence, those records after a big time gap with previous records have been dropped in our training dataset.

3.3 Information from nearby stations

In addition, we created attributes by other stations pollution measurements. For each station, we recorded the other closest stations based on their euclidean distance. Afterwards, few features in other closest stations (such as wind speed) with applying time shift algorithm are considered as new attributes. .

3.4 Arithmetic operation

Besides, arithmetic operation is applied in transforming the pollution records.

3.4.1 Percentage change

The percent change of pollution measurements of last two hours is defined as (pollution(t-1) - pollution(t-2))/pollution(t-2).

3.4.2 Feature combination

Last but not least, we created new attributes by multiplying different previous pollution measurements.

4 Feature transformation & other preparation

4.1 drop duplicates

There are some duplicates in raw data, that is to say, the same air pollution station have more than one record at the same time, so duplicates have been dropped by pandas.DataFrame.drop_duplicates.

4.2 outlier analysis

In order to remove outliers, we used pandas.DataFrame.quantile method with threshold equal to 3IQR, which means that the target value larger than Q3+3(IQR) will be pruned. However, it is found that pruning the outlier can only reduce the error in training. In the unseen data, outlier still exists.

4.3 handling with categorical data

Most Machine Learning algorithms prefer to work with numerical values, so we converted these categorical variables to ordinal by our custom class CategoryToOrdinal() method. We transformed categorical data into ordinal numbers (ranked by mean value versus the target variable) instead of dummy variable, avoiding the curse of dimensionality.

4.4 feature scaling

Converting all the features to standard score will accelerate the model training process, if the models involves gradient descent. In our potential ML algorithms, we standardize our data as z-score. Noted that for some algorithms like neural networks, it is advised to use zero-one scaling because neural networks often expect an input value ranging from 0 to 1.

4.5 dimension reduction

This is an optional transformation. PCA can speed up the model training process without losing much information. However, it is found that it would not improve the model accuracy since PCA does not provide new information.

5 Feature Selection

5.1 Feature exploration by visualization

Bar charts are drawn to illustrate the relationships between the input variables with the targets (there are three targets here). All images are saved in Graph folder. Take variable hour and target O3 as an example (Fig.3), it is shown that there is an obvious linear correlation between variable hour and target O3, where the x-axis represents the hour and y-axis represents the mean of O3 concentration according to hour.

5.2 Feature selection using recursive feature elimination(RFE)

Feature selection help finding the impact of each attributes in predicting the target. Recursive feature elimination(RFE) was adopted here, which selects the features by recursively considering smaller and smaller sets of features based on their feature importance. First, the estimator is trained on the initial set of features and the importance of each feature is obtained. Then, the least important feature is pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to

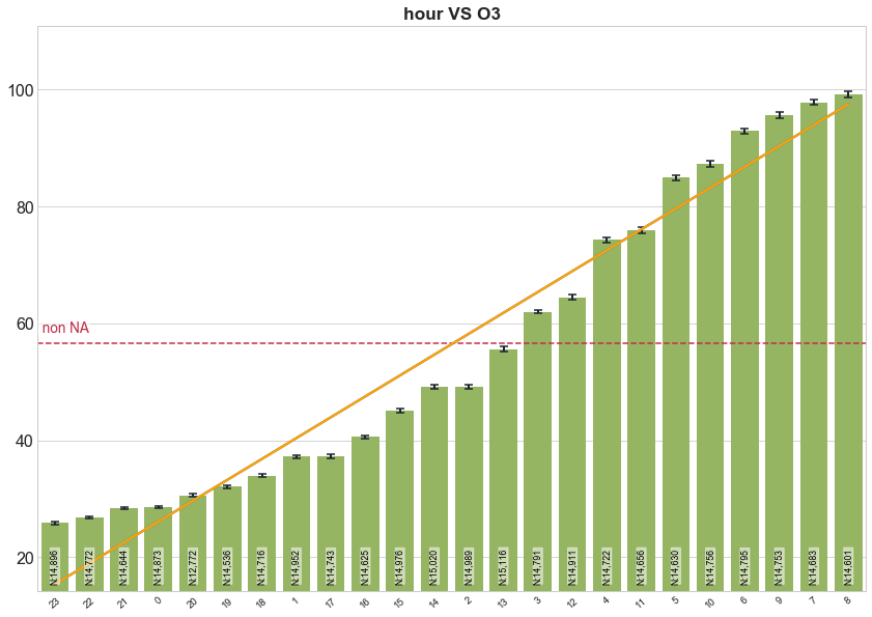

Figure 3

select is eventually reached. Apart from heavy run time, RFE is better than andomized decision trees or L1-based feature selection because recursive elimination can handle the the correlated feature better and we can understand the model performance with number of features applied.

In our project, we wrote a custom python script feature_selection_by_rfe to generate it with plotting, showing the relationship between model performance and number of features chosen. Finally, the result was saved as pickle files for record. The RFE results of three targets are shown as Fig.4, Fig.5, Fig.6:

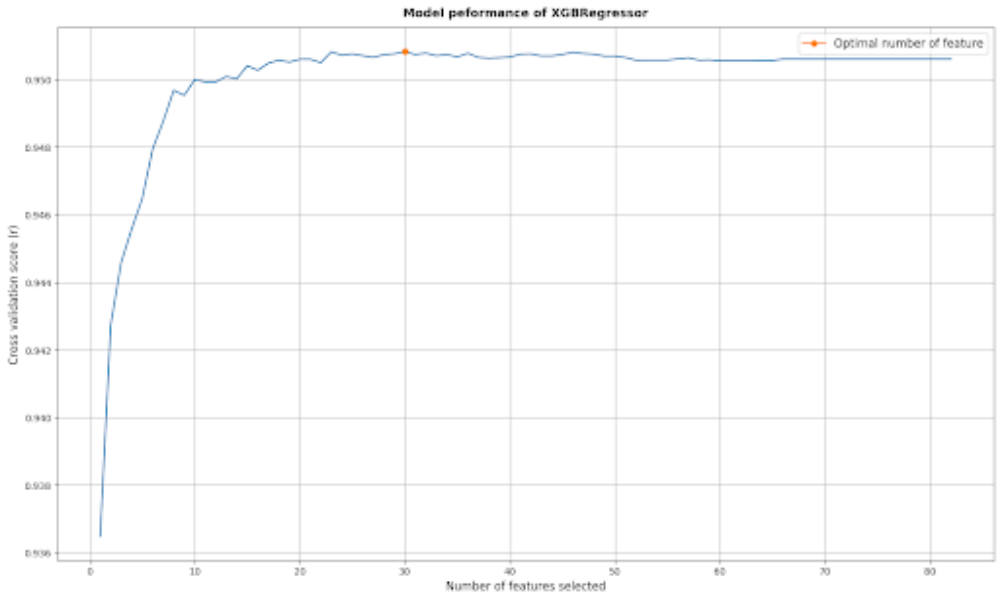

Figure 4

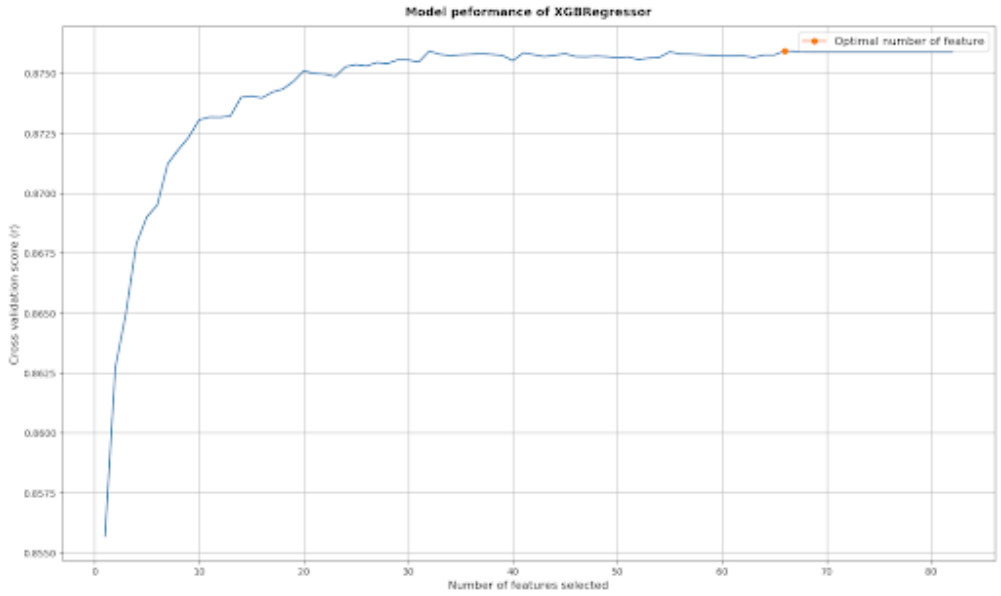

Figure 5

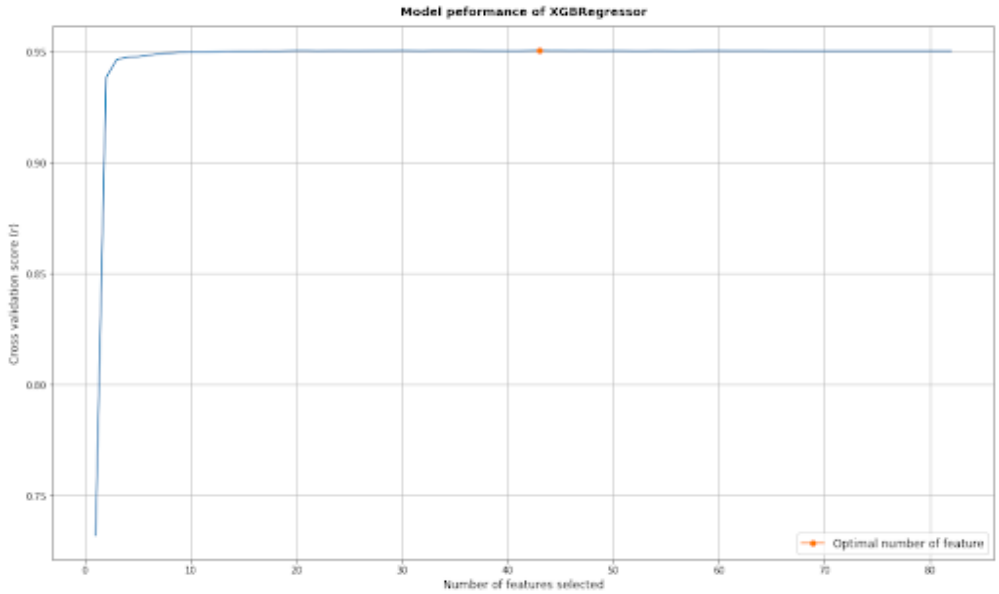

Figure 6

We also explored different ML algorithms such as Random-Forest (RF), but they are less accurate obviously (In Fig.7).

6 Model Development

6.1 Sampling strategy

Our data contains tremendous amount of record (more than 300,000 records), it takes time in performing RFE and hyperparameters tuning for a whole set of data. Hence a fraction of

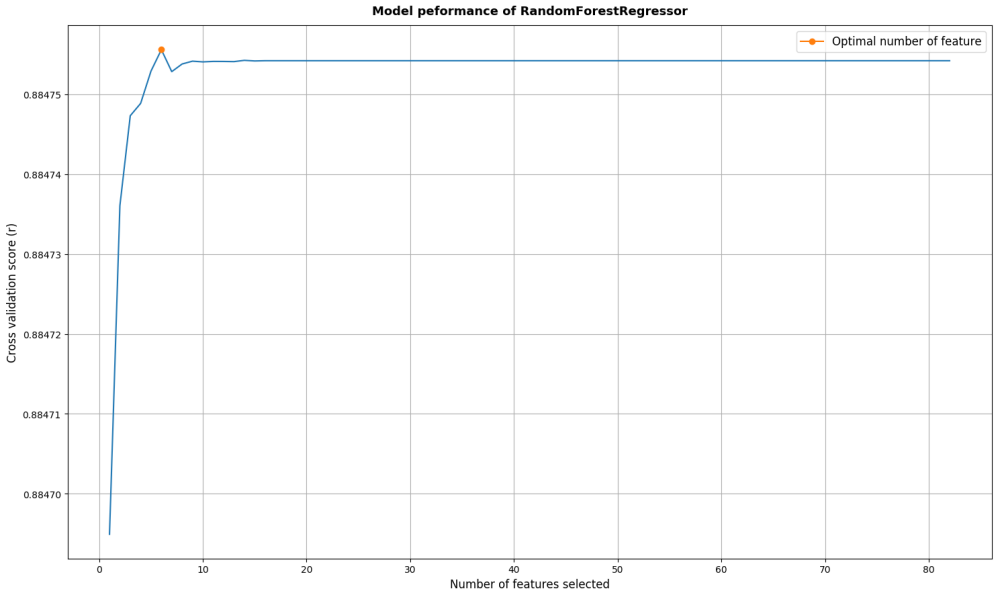

Figure 7

data is selected for RFE and hyperparameters tuning. Considering to predict the pollution on 1st and 2nd May 2018, we increase the sampling weight for those records are closer to 1st and 2nd May 2018. In training the whole dataset, the sample_weight is applied instead (which penalize more with higher sample_weight), in order to focus on the records closer to 1st and 2nd May 2018.

It was found that the April 2018 data is adequate to perform model development, so RFE and model development is based on the last month only instead.

6.2 Hyperparameters tuning

After preparing data for the machine learning algorithms, we choose some promising models and tune hyperparameters by using sklearn.model_selection.GridSearchCV. In response to the project request, hyperparameter is selected based on smape score with sklearn.metrics.make_scorer, under cross-validation.

6.2.1 XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. now we tune the hyperparameter of XGboost with a subset of data by stratified 3 folds of cross-validation. Parameters to be tuned include:

- subsample: subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees.
- reg_lambda: L2 regularization term on weights. Increasing this value will make model more conservative.

- reg_alpha: L1 regularization term on weights. Increasing this value will make model more conservative.

For target PM2.5, the best set of hyperparameters is subsample to 0.3, 'reg_alpha' to 0.5, 'reg_lambda' to 0.3.

For target PM10, we obtain the best solution by setting the subsample to 0.5, 'reg_alpha' to 0.3, 'reg_lambda' to 0.2.

For target O3, we obtain the best solution by setting the subsample to 0.5, 'reg_alpha' to 0.4, 'reg_lambda' to 0.2.

6.2.2 LightGBM

Light gbm is a fast, distributed, high performance gradient boosting (GBDT, GBRT, GBM or MART) framework based on decision tree algorithms, it can be used for ranking, classification and many other machine learning tasks.

Noted that there are few changes in training lightgbm model. It is found it is hard to predict the next hour O3 by the previous O3 values, the air quality measurements of the previous hours are dropped from the feature sets, taking into account that the error will accumulate if we use our prediction results as input attributes.

The hyperparameters of LightGBM are tuned with a subset of data by 5 folds of cross-validation.

Parameters we tuned include:

- max_depth: It describes the maximum depth of tree. This parameter is used to handle model over-fitting.
- feature_fraction: Used when your boosting(discussed later) is random forest. 0.8 feature fraction means LightGBM will select 80 percent of parameters randomly in each iteration for building trees.
- bagging_fraction: specifies the fraction of data to be used for each iteration and is generally used to speed up the training and avoid over-fitting.
- num_leaves: number of leaves in full tree, default: 31
- reg_alpha: L1 regularization term on weights. Increasing this value will make model more conservative.
- reg_lambda: L2 regularization term on weights. Increasing this value will make model more conservative.

For target PM2.5, we obtain the best solution by setting the max_depth to 14, 'feature_fraction'to 0.5, 'bagging_fraction' to 0.3, 'num_leaves' to 60, 'reg_alpha' to 0.6, 'reg_lambda' to 0.9.

For target PM10, we obtain the best solution by setting the max_depth to 11, 'feature_fraction'to 0.7, 'bagging_fraction' to 0.3, 'num_leaves' to 150, 'reg_alpha' to 0.4, 'reg_lambda' to 0.2.

For target O3, we obtain the best solution by setting the max_depth to 10, 'feature_fraction'to 0.5, 'bagging_fraction' to 0.5, 'num_leaves' to 100, 'reg_alpha' to 0.2, 'reg_lambda' to 0.1.

6.3 Final model building

After selecting the best set of hyperparameters, now we will re-train the full set of data with tuning the hyperparameters to avoid over-fitting. The final evaluation showed the r-square and smape score in cross-validation. The results are shown in table1 and table2.

Table 1: Cross Validation Results(XGBoost)

target	train_mean_r2	test_mean_r2	train_mean_smape	test_mean_smape
PM2.5	0.9689	0.9532	0.1638	0.1787
PM10	0.8592	0.8155	0.1676	0.1754
O3	0.9683	0.9538	0.2214	0.2219

Table 2: Cross Validation Results(LightGBM with some features dropped)

target	train_mean_r2	test_mean_r2	train_mean_smape	test_mean_smape
PM2.5	0.9105	0.8026	0.2940	0.3703
PM10	0.8561	0.6581	0.1896	0.2848
O3	0.9396	0.8150	0.2543	0.3541

6.4 Last two days validation

In order to further validate the model performance, the models are applied to predict air quality measurements on further validation, the last 2 training days in 29th and 30th April 2018, and accuracy is measured by smape.

The results are shown in table3. It is observed that smape gets much larger, almost twice as the result of cross validation, while actual values and prediction values are compared by scatterplots.(Fig.8) Our model is able to predict successfully the trend and fluctuation of the pollution. There is some rare pollution having rapid change and extremely high values, which is highly hard to capture in machine learning.

Based on the last two days validation results on XGboost and LightGBM, it is found that LightGBM is superior than XGBoost in predicting O3. Hence, in the final submission.csv, the O3 prediction is replaced by LightGBM while PM2.5 and PM10 remain the same.

Table 3: Prediction results of two models)

model	smape of PM2.5	smape of PM10	smape of O3
XGBoost	0.4538	0.5748	0.7733
LightGBM	0.6943	0.5578	0.3891

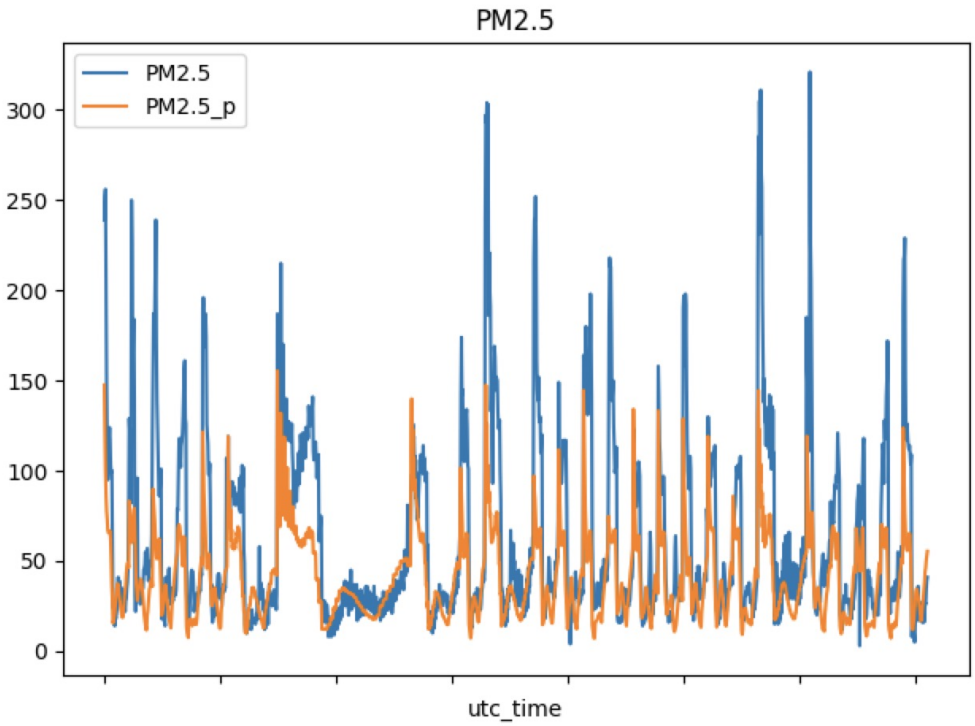

Figure 8

7 Conclusion

- XGBoost model works well in cross validation, but doesn't work well in predicting long term air quality. Because air quality measurements in the previous hours are actual values in model development. However, air quality measurements in the previous hours are our prediction results in predicting 1st and 2nd May 2018, which means error will accumulate. Therefore, it is advised to use this model only when you are required to predict air quality for few hours.
- Machine learning algorithms is limited to predict rapid change of air quality measurements (outlier).

8 Task Assignment

Data integration: Wang Chuhan

Feature Engineering: Hui Kwat Kong

Model Developing: Hui Kwat Kong, Wang Chuhan

Writing Report: Wang Chuhan, Hui Kwat Kong