Analysis of PM2.5 data in Beijing

May 25, 2022

1. Background Introduction

PM2.5 consists of airborne particles with aerodynamic diameters of less than 2.5 μm. They are known to influence visibility, human health and even climate. Beijing and a substantial part of China are experiencing chronic air pollution, PM2.5 in particular. Epidemiological evidence shows that exposure to PM2.5 can cause lung morbidity, serious respiratory and cardiovascular diseases, and even death.

Therefore, it is meaningful to analyze PM2.5 data in Beijing. Our analysis uses hourly PM2.5 readings taken at the US Embassy in Beijing located at (116.47 E, 39.95 N), in conjunction with hourly meteorological measurements at Beijing Capital International Airport (BCIA), obtained from weather.nocrew.org. Both data series run from 1 January 2010 to 31 December 2014. Our study combines PM2.5 and meteorological data for an extended time span (5 years) in studying China's PM2.5 pollution, and we added some time variables such as seasons, mornings and evenings and so on.

2. Data statistics and visualization

Before we build the model to train the data and do some predictions, we firstly conduct statistics and visualization of the data. From this, we tried to directly explore the general change rule of PM2.5 concentration in Beijing from the visualization chart, as well as

the significant influence of some characteristics such as different wind directions on PM2.5 concentration.

Firstly, we deal with the PM2.5 concentration information of each year and made the following line graph (Figure 1) of averages and medians of PM2.5 concentration in five years. From the graph, we can find that Beijing's PM2.5 concentration has generally shown a slight decline, except for a large increase in 2013. This is consistent with the fact that the Chinese government has taken note of this and in late 2013 issued a new policy on air pollution and specified a series of energy-saving and emission reduction plans in the hope of reducing PM2.5 concentration.

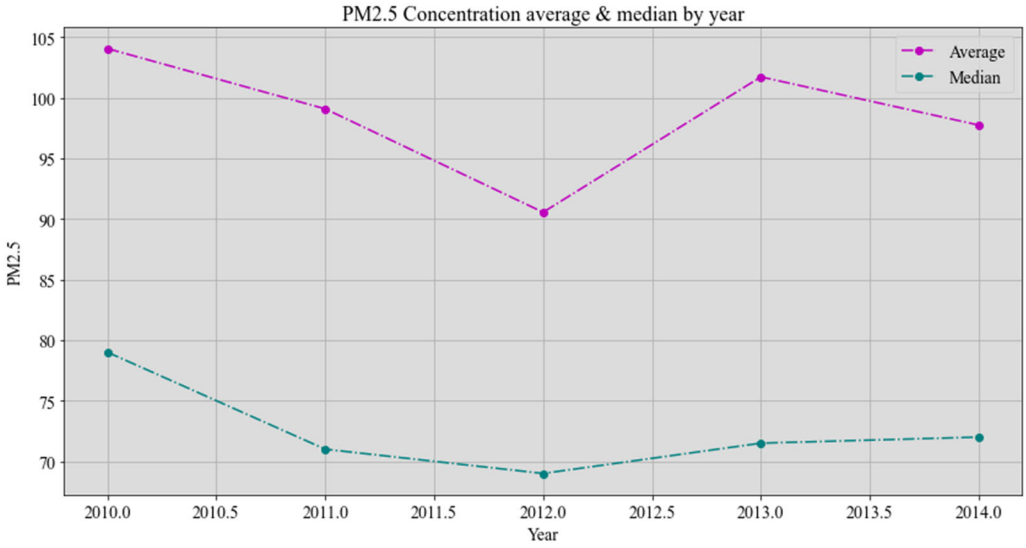

Figure 1

To be more precise, we made a line chart (Figure 2) of the monthly mean and median changes of PM2.5 concentration with each month. From the picture, we can clearly see that PM2.5 concentration is generally not high in summer (for Beijing) every year, while the months with high PM2.5 concentration almost all occur in winter in Beijing. One important reason is that the current situation of collective heating in winter in northern China greatly increases the content of air pollutants, including PM2.5. We will also take this into account in future modeling (in task2).

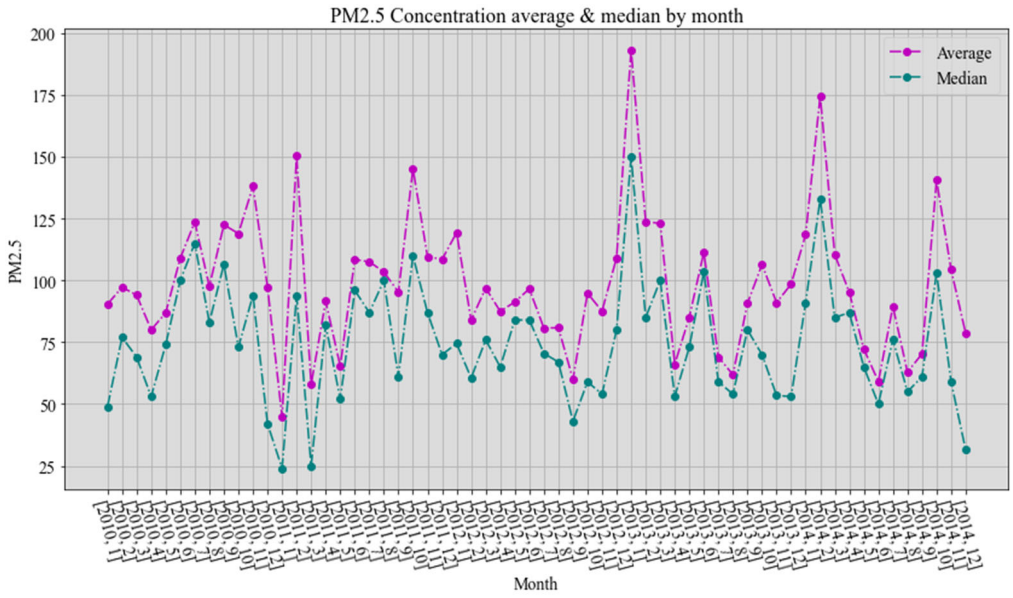

Figure 2

We partition the PM2.5 time series into three states: low PM state when PM2.5 ≤ $35\mu g/m^3$; medium PM state when PM2.5 > 35 $\mu g/m^3$ and PM2.5 ≤ 150 $\mu g/m^3$; and very high PM when PM2.5 > 150 $\mu g/m^3$. We classified PM2.5 concentration data of each year and month, and made folded bar charts (figure 3 and figure 4). From these two pictures, we can get results that are consistent with our previous observations.

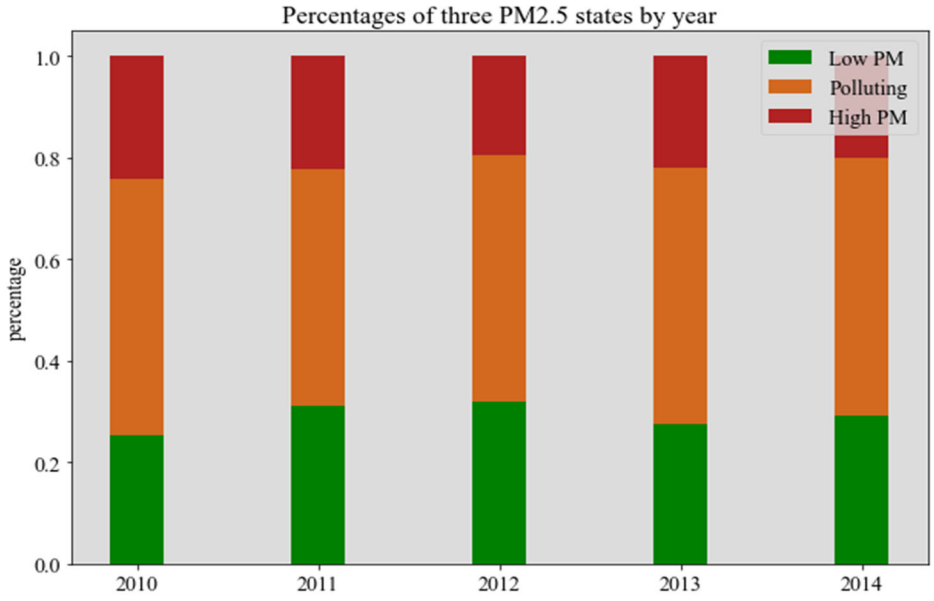

Figure 3

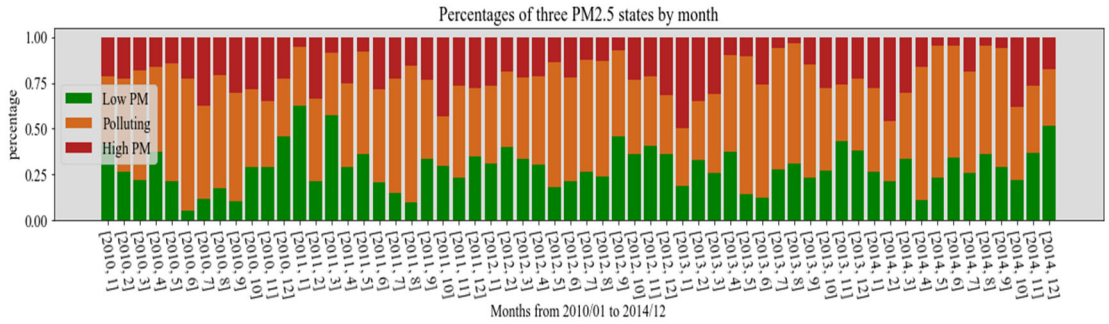

Figure 4

In order to intuitively understand the impact of each season as well as heating and farmers' harvest season on PM2.5, we drew the following folded bar chart (Figure 5). It is worth noting that the second month of autumn and the whole winter is the heating period in Beijing, and the first month of autumn is the harvest season for farmers. We can conclude that farmers' harvest and heating significantly increase the concentration of PM2.5 in the air. (Consider the weather in Beijing, we make the following agreement: 'Spring' means April to May; 'Summer' means June to September; 'Autumn(harvest)' means October; 'Autumn' means November; 'Winter' means December to March in the next year.)

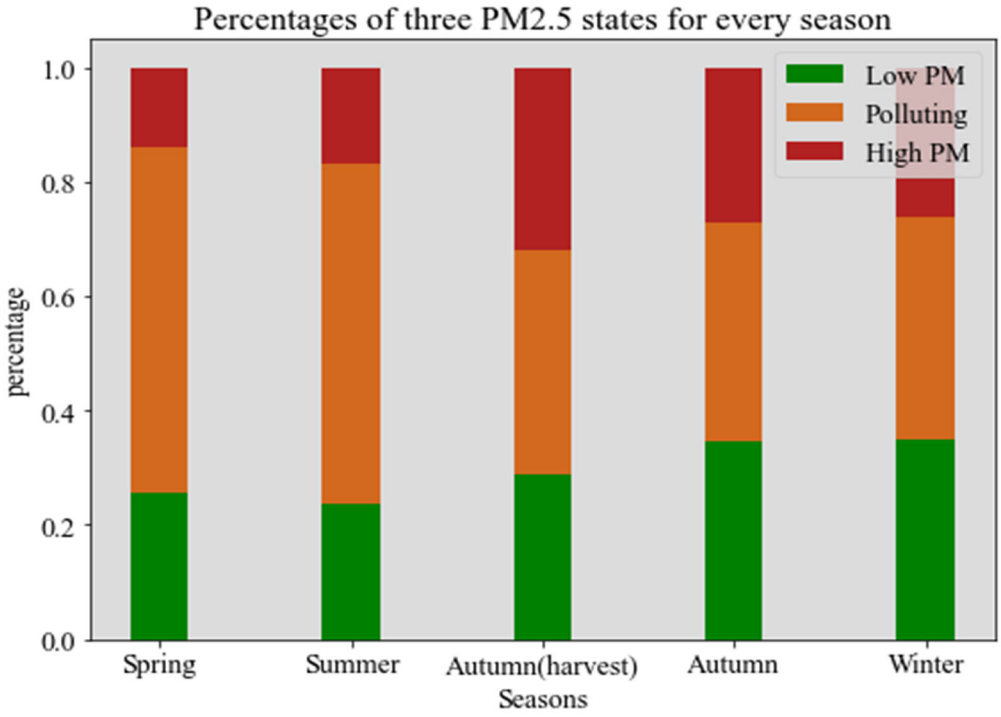

Figure 5

We noticed that wind direction is also one of the important factors affecting PM2.5 concentration. From the data set, the directions can be grouped into four broad categories: northwest (NW), which includes W, WNW, NW, NNW and N; northeast (NE), for NNE, NE and ENE; southeast (SE), covering E, ESE, SE, SSE and S; and calm variable (CV). We made a graph (Figure 6) of PM2.5 concentration under different wind directions and two graphs (Figure 7& Figure 8) of PM2.5 concentration under different seasons and wind directions. And it is obvious that in each season, the influence of wind direction on PM2.5 concentration is not negligible. For example, the concentration of PM2.5 is lower during windy periods than during calm periods. Therefore, in the subsequent modeling process, we will consider the wind directions as features in the whole process.

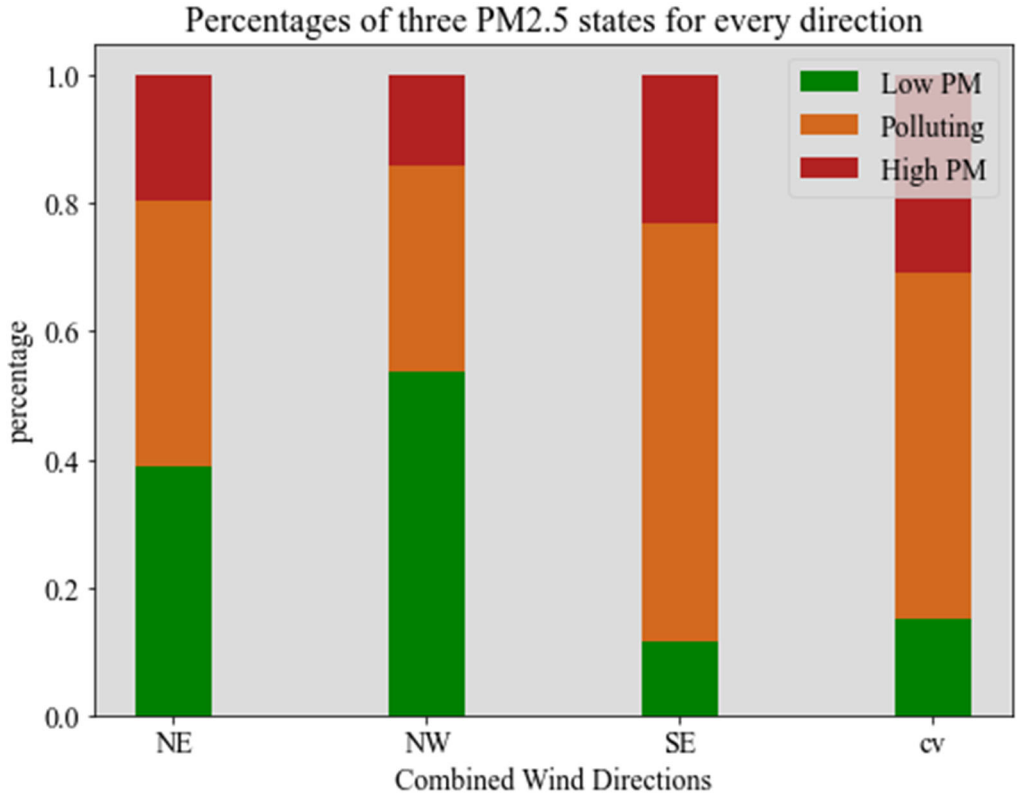

Figure 6

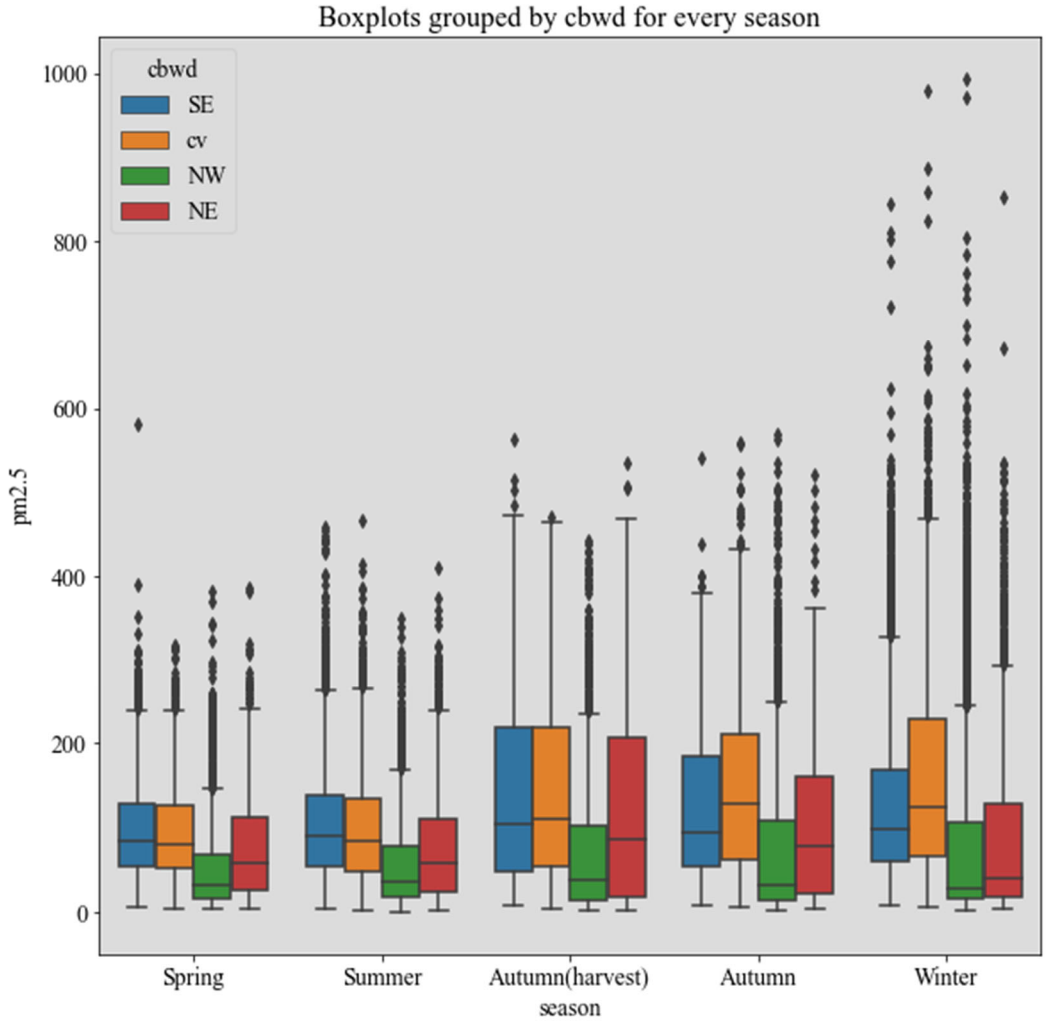

Figure 7

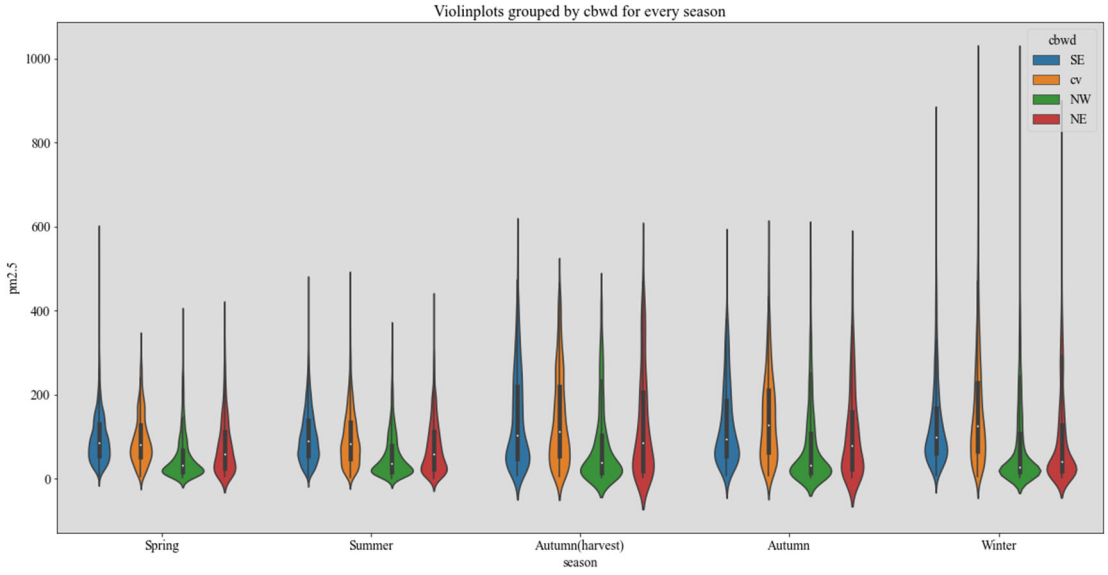

Figure 8

3. Task 1

Step 1: Introduce four features to deal with the feature 'cbwd' in the original data set. We call them 'cvornot', 'NEornot', 'NWornot' and 'SEornot'. If the word in ‘cbwd’ is ‘cv’, the corresponding 'cvornot' is 1, and other three variables are all 0. And similar things for other cases.

Step 2: Delete the samples with PM2.5 concentration ‘NA’ .

Step 3: Use LOF algorithm to identify outliers and remove them. In fact, we have about 40,000 samples, so we can delete some sample points in order to get a better predictive model.

Step 4: Split data to covariate vector X (contain 'DEWP', 'TEMP', 'PRES', 'Iws', 'Is', 'Ir', 'cvornot', 'NEornot', 'NWornot', 'SEornot' as features) and response y and do ‘log’ transformation to y value of each sample. Log transformation is a common data preprocessing method. We use it to make the data more stable and weaken the collinearity and heteroscedasticity of the model.

Remark: We tried some methods of data preprocessing, such as z-score, max-min transformation, box-cox transformation and log transformation. Finally, we found that the log transformation works best. In fact, the log transform helps to make the data values normally distributed, which is better to our models.

Step 5: Divide all data into training set and test set. We split the data to training and test sets by following this rule: extract the items (rows) for every 7-day from 2010-01-07 to 2014-12-25; that is, choose the items on 2010-01-07, 2010-01-14, 2010-01-21, 2010-01-28, 2010-02-04, 2010-02-11, ..., the last item is on 2014-12-25. Collect these items as the test set, and keep the remaining items as training set.

Step 6: Do the classical least squares regression. We train the model by training set after step 5, and compute the R-scores (omit it by ‘scores’ in the after paragraphs) over training set and test set. The result is:

```
(0.4710063630059297, 0.47525687275781114)
```

Remark: The first value is the score over training set and the second one is the score over test score. (This order is also followed in the following text, and we will not repeat

the narrate.)

In order to intuitively observe the predictive effect of the model on the test set, we draw the frequency distribution histogram of predicted y-value and actual y-value (Figure 9).

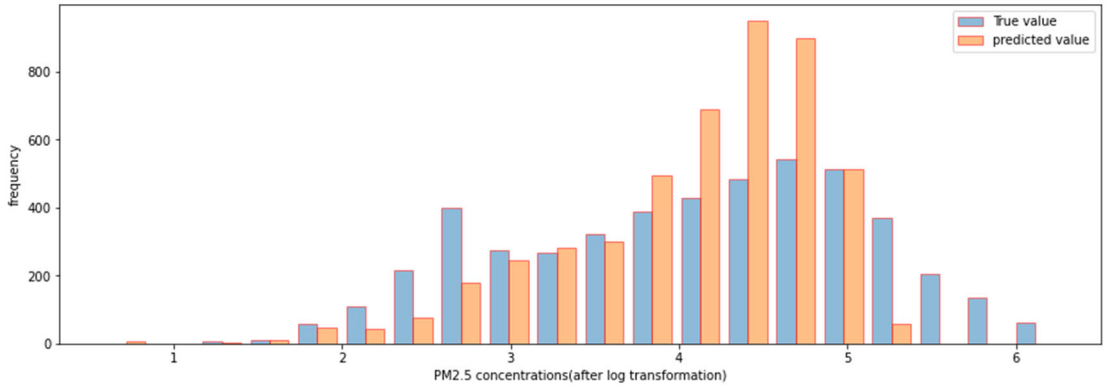

Figure 9

Step 7: Do the ridge regression with parameters determined by 'CV' method. We train the model by training set after step 5, and compute the R-scores. The result is:

```
(0.4710060579607478, 0.47524613613559497)
```

We draw the frequency distribution histogram of predicted y-value and actual y-value (Figure 10).

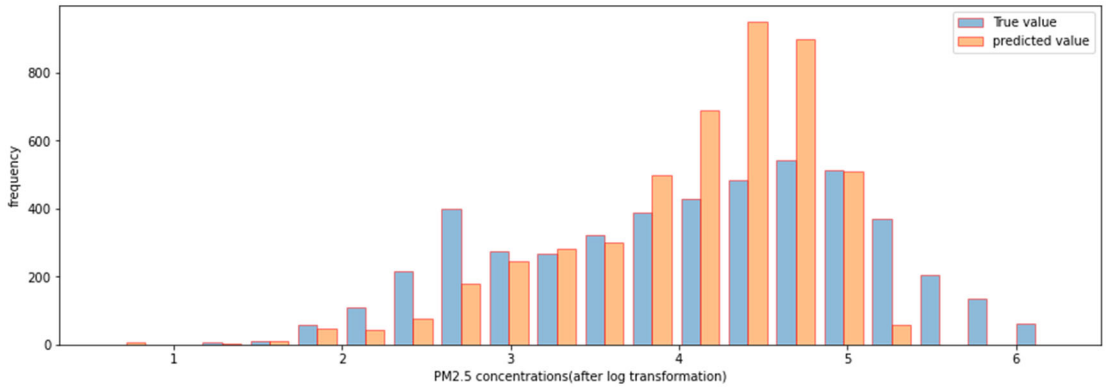

Figure 10

Step 8: Do the LASSO regression with parameters determined by 'CV' method. We train the model by training set after step 5, and compute the R-scores. The result is:

```
(0.47100522194421646, 0.47527106886156123)
```

We draw the frequency distribution histogram of predicted y-value and actual y-value

(Figure 11).

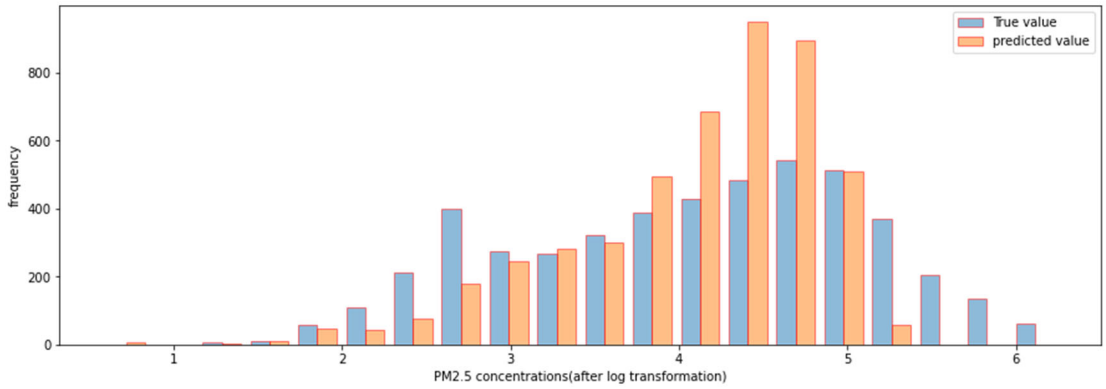

Figure 11

Step 9: Do XGBoosting regression (which is a common boosting regression method). According to the scores of steps 6 ~ 8, the effects of OLS, ridge and LASSO regression methods are not ideal. Therefore, we consider the ensemble models. In this step, we choose XGBoosting regression.

Remark: We import 'hyperopt' in python to choose the optimal parameter in XGB-regression model automatically in order to the higher score over the test set.

We train the model by training set after step 5, and compute the R-scores. The result is:

```
(0.6349812279158589, 0.6040126408011938)
```

We draw the frequency distribution histogram of predicted y-value and actual y-value (Figure 12).

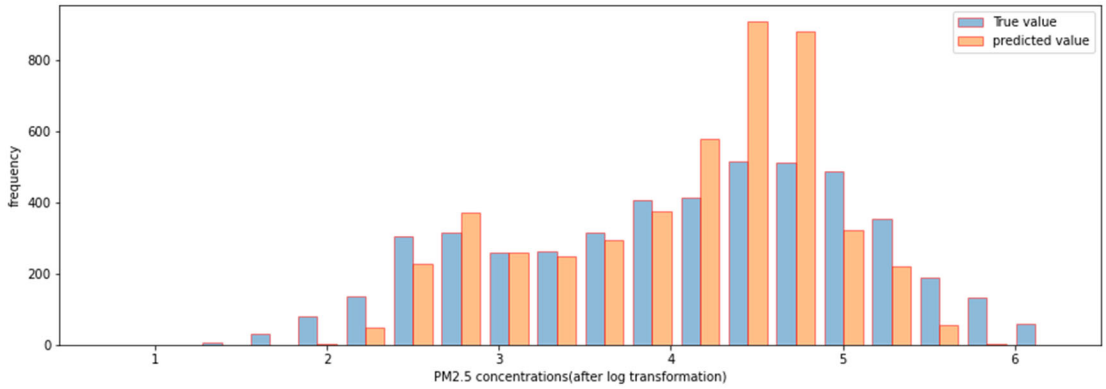

Figure 12

Step 10: Do Gradient Boosting Regression. We train the model by training set after step

5, and compute the R-scores (the method of selecting parameters is in the same way as the previous step). The result is:

```
(0.6323144242001061, 0.6036154800963645)
```

We draw the frequency distribution histogram of predicted y-value and actual y-value (Figure 13).

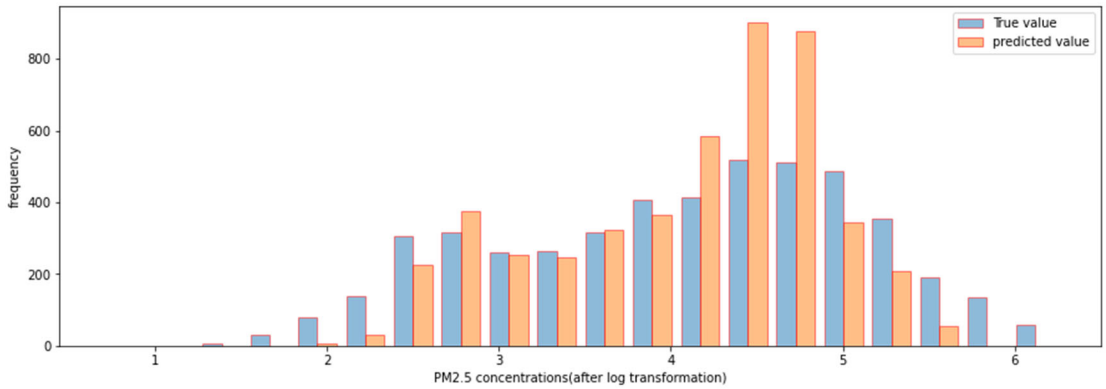

Figure 13

Step 11: Do Cat Boosting Regression. Cat Boosting, a machine learning library open-source by Russian search giant Yandex in 2017, is a Boosting algorithm. Cat Boosting and XGBoosting are both improved implementations under the framework of GBDT algorithm. Cat Boosting claims to be a better algorithm than XGBoosting and LightGBM in terms of algorithmic accuracy. We compute some scores of Cat Regression over test set under different value of parameter 'learning_rate'. The result is:

	learning_rate	score
0	0.039770	0.609357
1	0.039775	0.608113
2	0.039780	0.608471
3	0.039785	0.608297
4	0.039790	0.608564
5	0.039795	0.608467
6	0.039800	0.608530
7	0.039805	0.607953
8	0.039810	0.608324
9	0.039815	0.609629
10	0.039820	0.608913

And we let learning_rate equal to 0.039815, n_estimators=1500, max_depth=6 and keep the default values for other parameters. Then we train the model by training set after step 5, and compute the R-scores. The result is:

```
(0.6534002358869109, 0.6096294503565602)
```

We draw the frequency distribution histogram of predicted y-value and actual y-value (Figure 14).

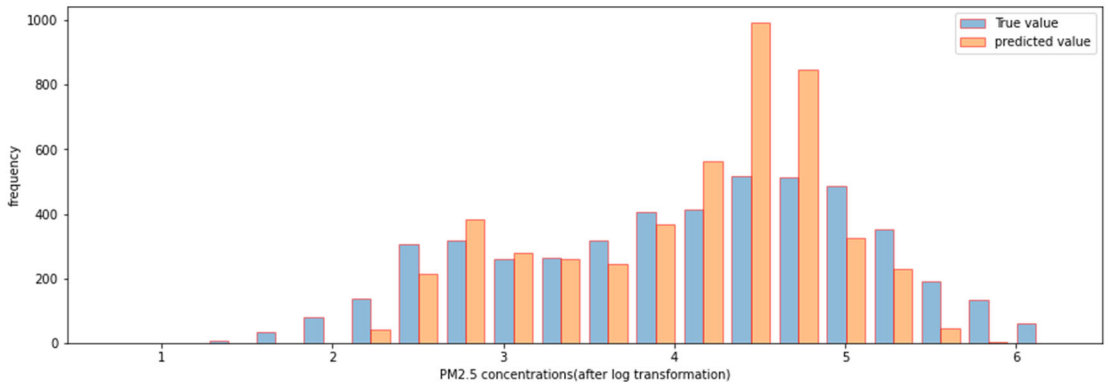

Figure 14

Summary of task 1: Above all, the Cat regression model we trained has the best score over test set.

4. Task 2

Step 1: Introduce four features to deal with the feature 'cbwd' in the original data set. We call them 'cvornot', 'NEornot', 'NWornot' and 'SEornot'. If the word in ‘cbwd’ is ‘cv’, the corresponding 'cvornot' is 1, and other three variables are all 0. And similar thing for other cases.

Introduce four features to deal with samples in different hours. They are 'nightornot', 'morningornot', 'afternoonornot' and 'eveningornot'. If the hour of a sample is between 0 to 5, we set the 'nightornot' of it is 1, and other three features are 0. If the hour of a sample is between 6 to 11, we set the 'morningornot' of it is 1, and other three features are 0. If the hour of a sample is between 12 to 17, we set the 'afternoonornot' of it is 1, and other three features are 0. If the hour of a sample is between 18 to 23, we set the 'eveningornot' of it is 1, and other three features are 0.

Introduce one feature called ‘weekdaysornot’. If the date of a sample is in weekends, or is during Labor Day, National Day or Spring Festival, we set the feature of the sample is 0, and let it be 1 else.

Introduce one feature called 'before_policy_or_not'. If the year of a sample is 2014, we set the feature of it by 0, and we let it be 1 else. We introduced this feature because China's State Council issued new measures and targets to combat air pollutants in October 2013. And consider a two-month lag effect, so we do this.

Introduce four features to deal with samples in different months. The are 'month_class_one_or_not', 'month_class_two_or_not', 'month_class_three_or_not', 'month_class_four_or_not', 'month_class_five_or_not'. On the basis of Beijing's four seasons, we consider farmers' harvest season and urban heating season. We divide the twelve months of a year by five classes: April to May (spring), June to September (summer), October (autumn, harvest), November(autumn, heating period) and December to March in the next year (winter, heating period). And we let the four values of these four features be 1 (s) or 0 (s) , respectively by the same rules given before.

Step 2: Delete the samples with PM2.5 concentration ‘NA’ (Same operations as task 1).

Step 3: Use LOF algorithm to identify outliers and remove them (Same operations as

task 1).

Step 4: Split data to covariate vector X (contains 'DEWP', 'TEMP', 'PRES', 'Iws', 'Is', 'Ir', 'cvornot', 'NEornot', 'NWornot', 'SEornot', 'nightornot', 'morningornot', 'afternoonornot', 'eveningornot', 'weekdaysornot', 'before_police_or_not', 'month_class_one_or_not', 'month_class_two_or_not', 'month_class_three_or_not', 'month_class_four_or_not', 'month_class_five_or_not' as features) and response y and do 'log' transformation to y value of each sample (Same operations as task 1).

Step 5: Divide all data into training set and test set (Same operations as task 1).

Step 6: Do the classical least squares regression. We train the model by training set after step 5, and compute the R-scores (omit it by 'scores' in the after paragraphs) over training set and test set. The result is:

```
(0.5602751098902001, 0.5443339535991123)
```

In order to intuitively observe the predictive effect of the model on the test set, we draw the frequency distribution histogram of predicted y-value and actual y-value (Figure 15).

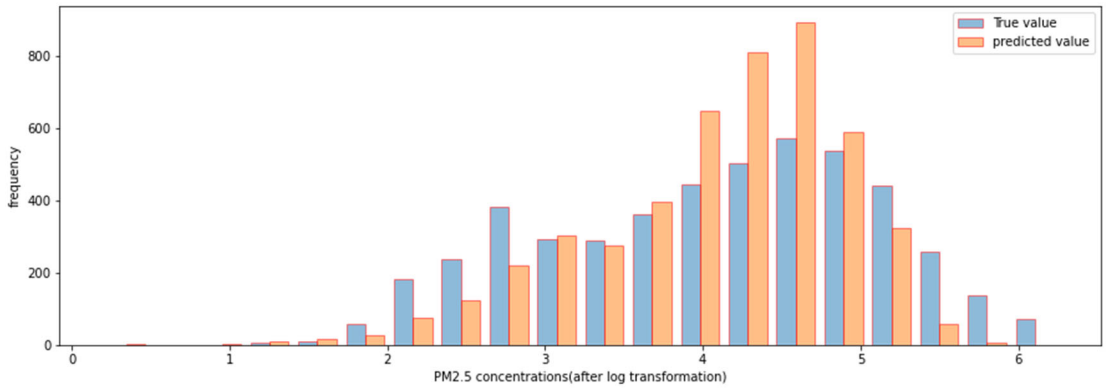

Figure 15

Step 7: Do the ridge regression with parameters determined by 'CV' method. We train the model by training set after step 5, and compute the R-scores. The result is:

```
(0.560274973089837, 0.5443658602840478)
```

We draw the frequency distribution histogram of predicted y-value and actual y-value (Figure 16).

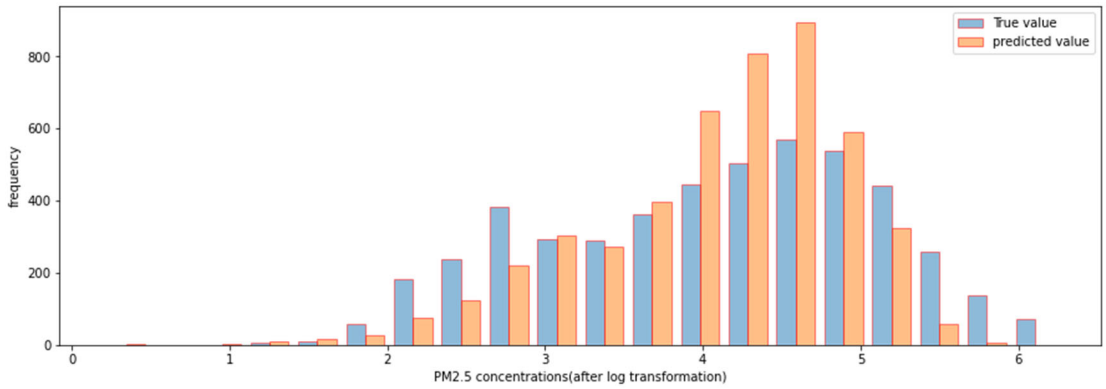

Figure 16

Step 8: Do the LASSO regression with parameters determined by 'CV' method. We train the model by training set after step 5, and compute the R-scores. The result is:

```
(0.5577392425490841, 0.5458571481917911)
```

We draw the frequency distribution histogram of predicted y-value and actual y-value (Figure 17).

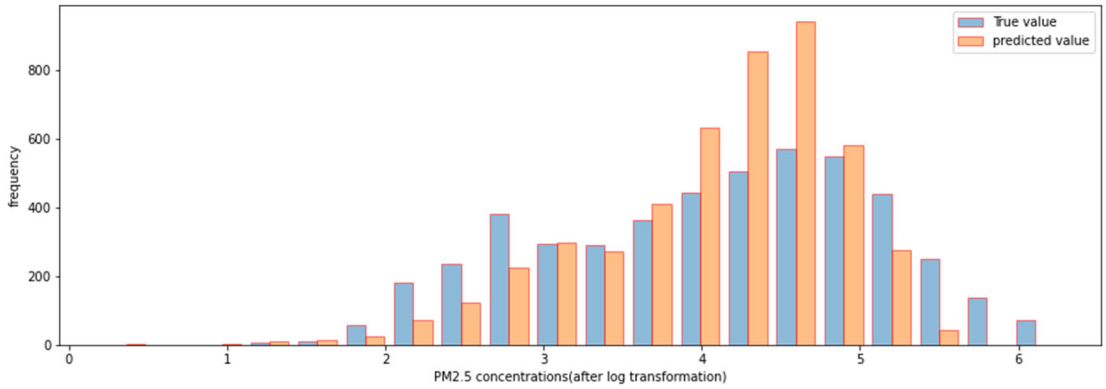

Figure 17

Step 9: Do XGB-regression. According to the scores of steps 6 ~ 8, the effects of OLS, ridge and LASSO regression methods are not ideal. Therefore, we consider the ensemble models. In this step, we choose XGB-regression.

Remark: We import 'hyperopt' in python to choose the optimal parameter in XGB-regression model automatically in order to the higher score over the test set.

We train the model by training set after step 5, and compute the R-scores. The result is:

```
(0.7144390761571212, 0.6503456343548257)
```

We draw the frequency distribution histogram of predicted y-value and actual y-value (Figure 18).

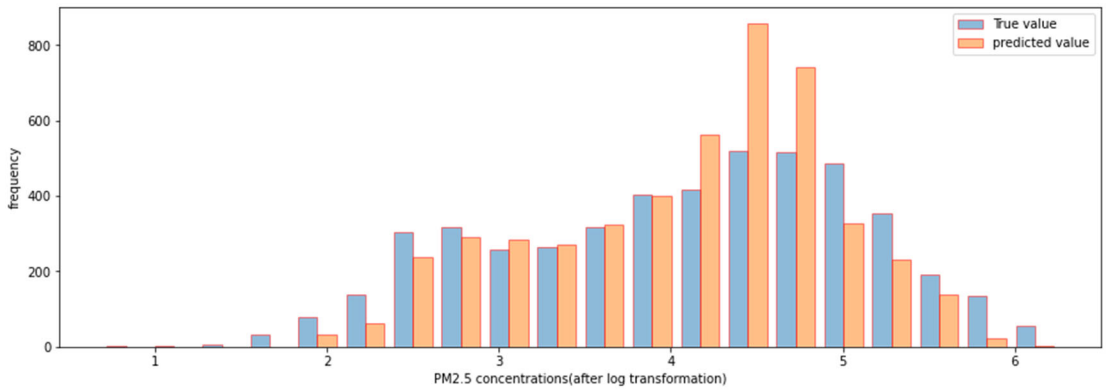

Figure 18

Step 10: Do Gradient Boosting Regression. We train the model by training set after step 5, and compute the R-score (the method of selecting parameters is in the same way as the previous step). The result is:

```
(0.714062106331045, 0.6514032865247327)
```

We draw the frequency distribution histogram of predicted y-value and actual y-value (Figure 19).

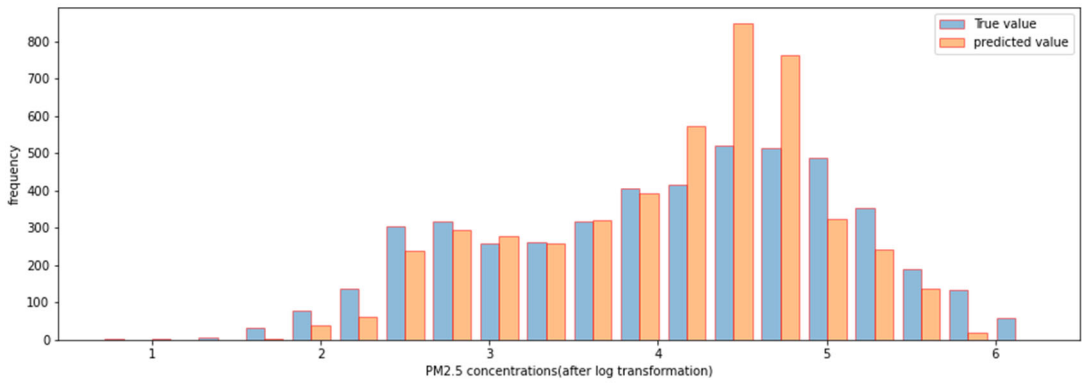

Figure 19

Step 11: Do Cat Regression. We compute some scores of Cat Regression over test set under different value of parameter 'learning_rate' and compare them. And we let learning_rate equal to 0.039815, n_estimators=1500, max_depth=6 and keep the default values for other parameters. Then we train the model by training set after step

5, and compute the R-scores. The result is:

```
(0.7455763356874812, 0.6562128895314473)
```

We draw the frequency distribution histogram of predicted y-value and actual y-value (Figure 20).

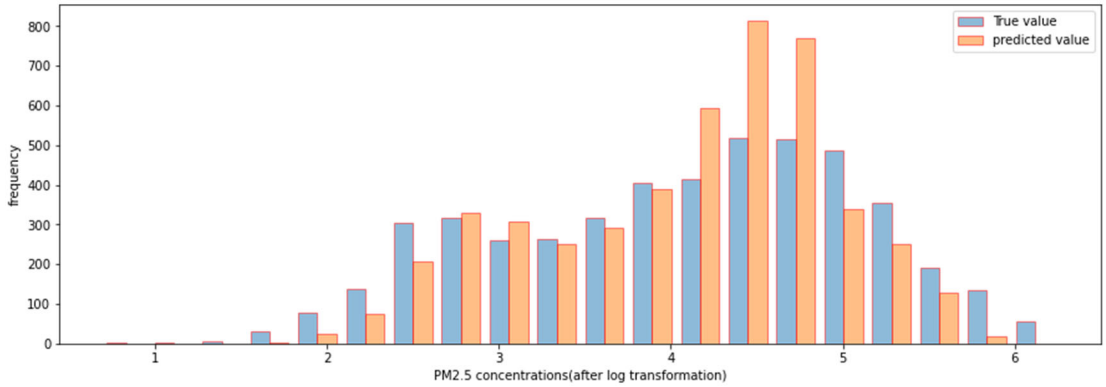

Figure 20

Summary of task 2: Above all, the Cat regression model we trained has the best score over test set.

5. Task 3

5.1 Fill the missing-value and do the same things as task 1

In the step 2 of task 1, we just delete the samples with missing data. Now, we select three methods to import the missing-value: average filling (denoted by filling method 1), kNN filling (denoted by filling method 2) and filling with the prediction value of an extreme forest regression model trained by the data without missing data (denoted by filling method 3).

Under each filling method, we do the same things as task 1 (except step 2), and we write the result here.

5.1.1 Fill the missing-value with filling method 1

Score of OLS:

```
(0.4474176874117457, 0.447694557958876)
```

Effect of OLS (figure 21):

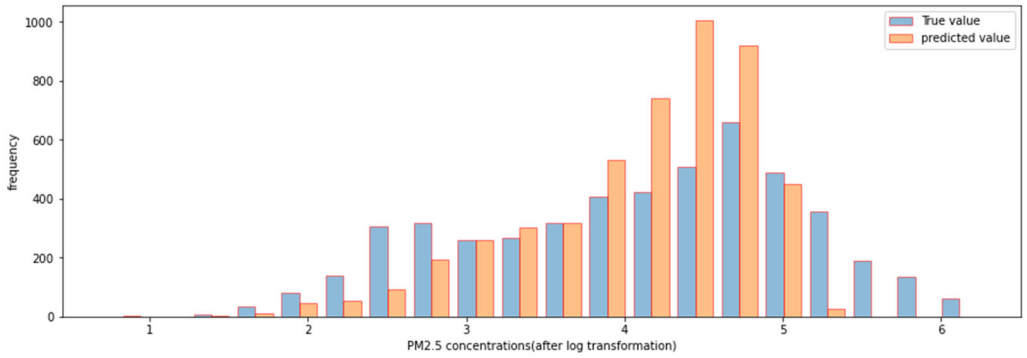

Figure 21

Score of ridge regression (with CV):

```
(0.4474172579692066, 0.4476865764305409)
```

Effect of ridge regression (with CV) (figure 22):

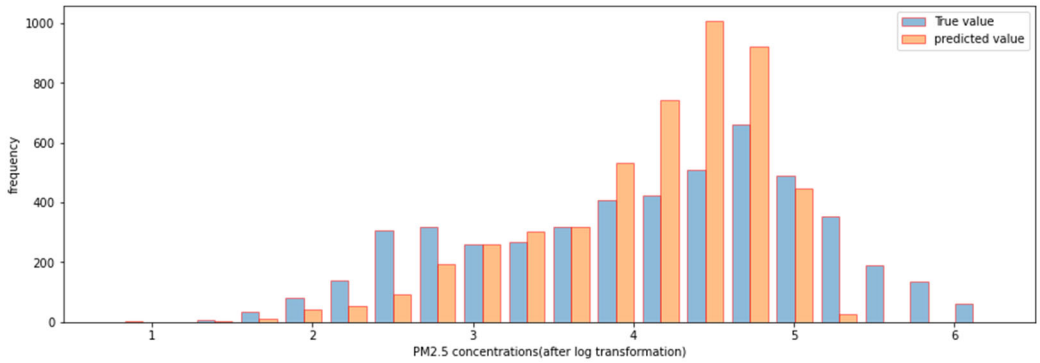

Figure 22

Score of LASSO regression (with CV):

```
(0.447384191594277, 0.4478134104671877)
```

Effect of LASSO regression (with CV) (figure 23):

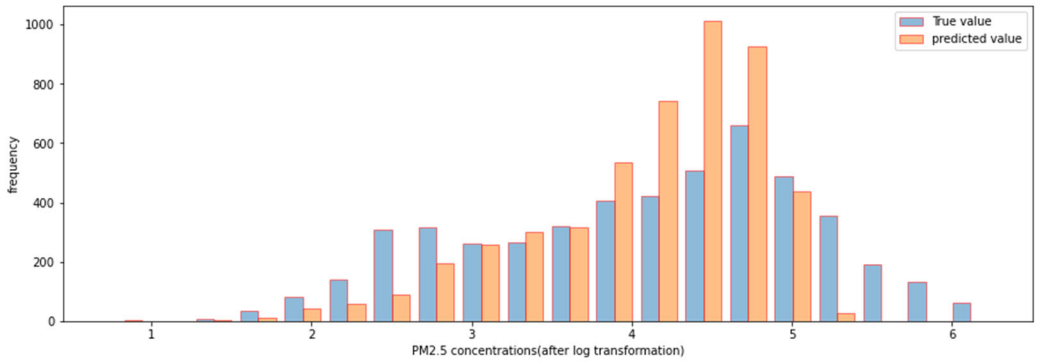

Figure 23

Score of XGBoosting regression:

```
(0.6039168100914913, 0.5732824986130107)
```

Effect of XGBoosting regression (figure 24):

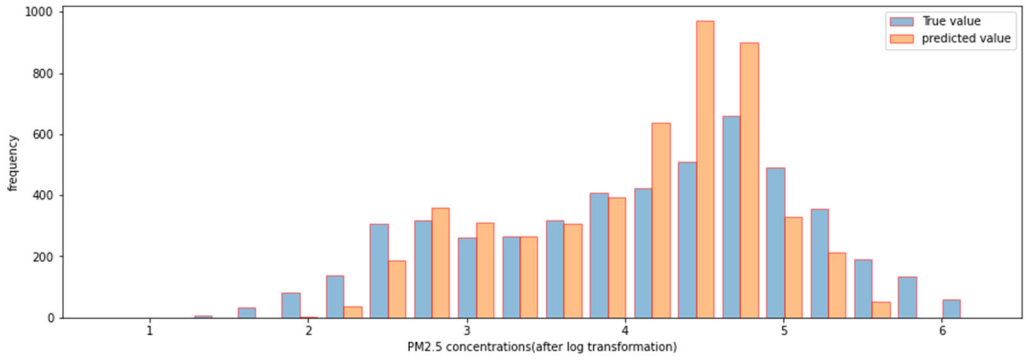

Figure 24

Score of Gradient Boosting Regression:

```
(0.6075296131887689, 0.573321206676253)
```

Effect of Gradient Boosting Regression (figure 25):

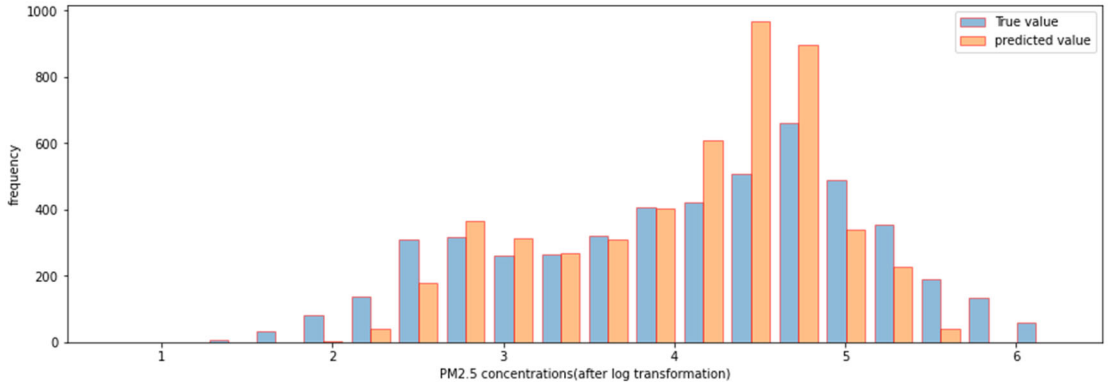

Figure 25

Score of Cat Regression:

```
(0.6287170629381031, 0.5778102983791574)
```

Effect of Cat Regression (figure 26):

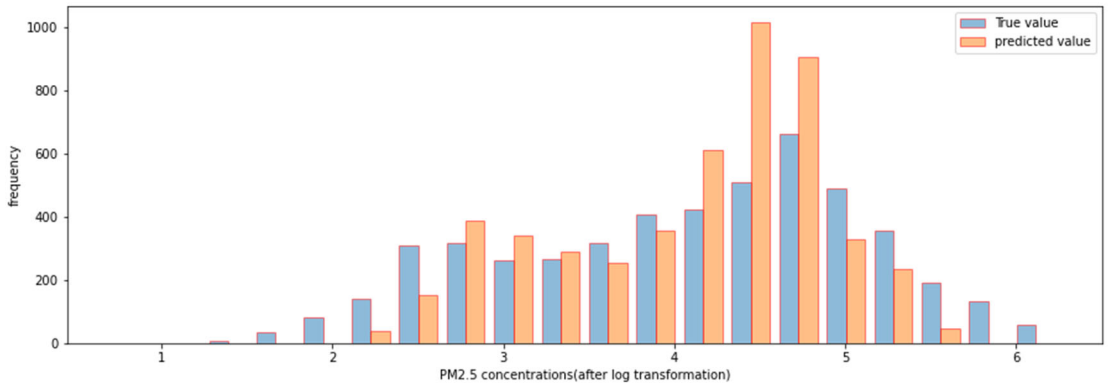

Figure 26

5.1.2 Fill the missing-value with filling method 2

Score of OLS:

```
(0.4746220624953863, 0.4773523367676177)
```

Effect of OLS (figure 27):

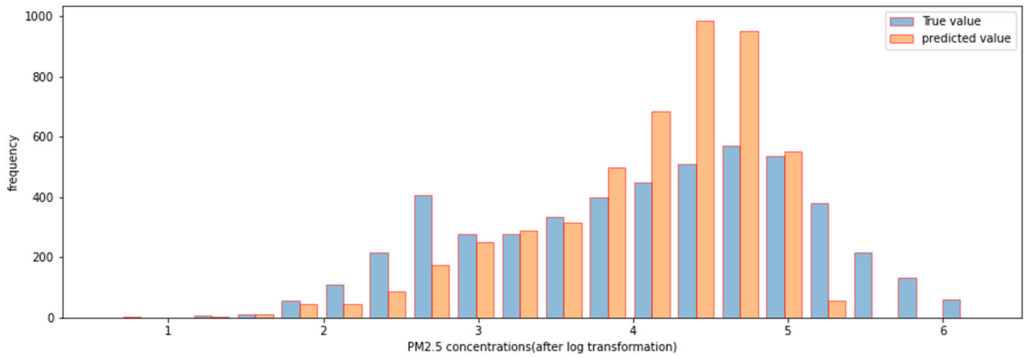

Figure 27

Score of ridge regression (with CV):

```
(0.4746216410381413, 0.4773396856627422)
```

Effect of ridge regression (with CV) (figure 28):

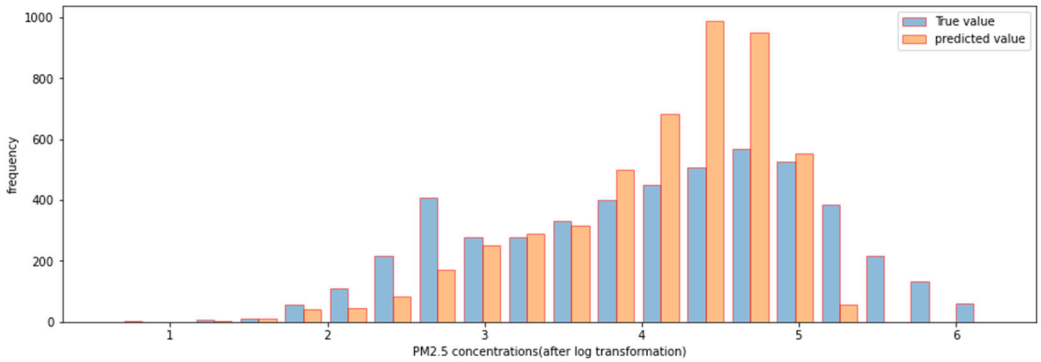

Figure 28

Score of LASSO regression (with CV):

```
(0.47460935294836426, 0.4773909532932814)
```

Effect of LASSO regression (with CV) (figure 29):

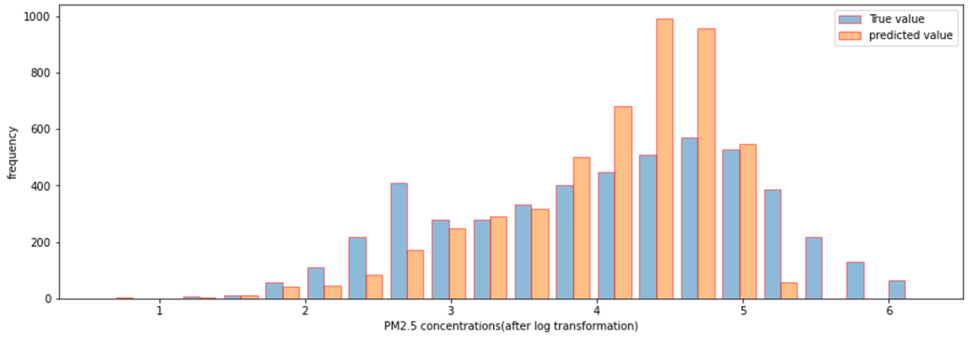

Figure 29

Score of XGBoosting regression:

```
(0.6314785532742326, 0.604519871298155)
```

Effect of XGBoosting regression (figure 30):

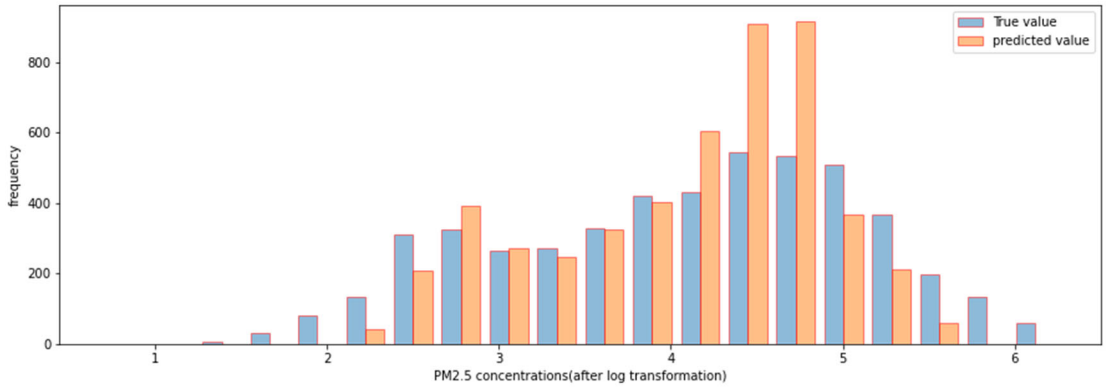

Figure 30

Score of Gradient Boosting Regression:

```
(0.6342409242652443, 0.604499928520823)
```

Effect of Gradient Boosting Regression (figure 31):

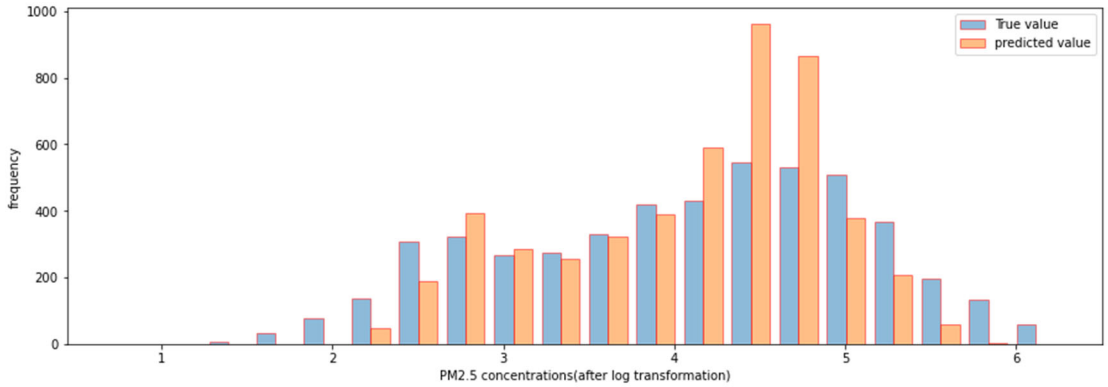

Figure 31

Score of Cat Regression:

```
(0.6555512993792623, 0.609796896106584)
```

Effect of Cat Regression (figure 32):

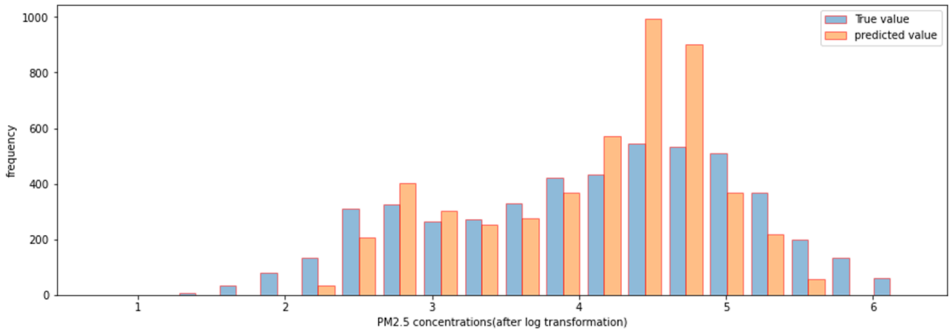

Figure 32

5.1.3 Fill the missing-value with filling method 3

Score of OLS:

```
(0.47538683253786207, 0.4784991310831693)
```

Effect of OLS (figure 33):

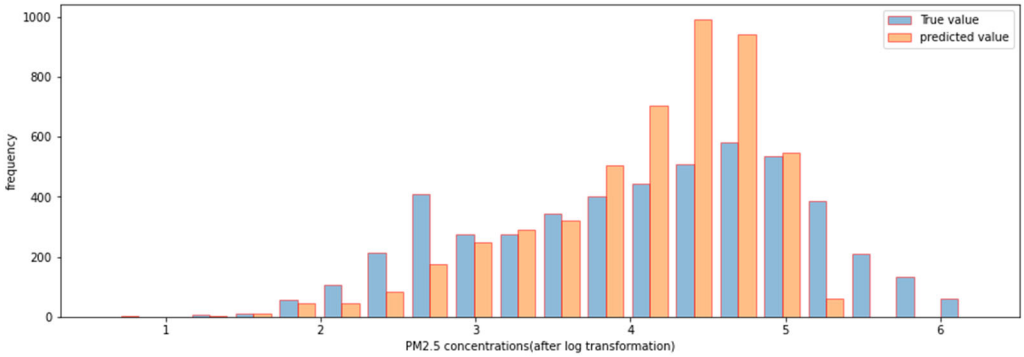

Figure 33

Score of ridge regression (with CV):

```
(0.4753863892217086, 0.47848430508095585)
```

Effect of ridge regression (with CV) (figure 34):

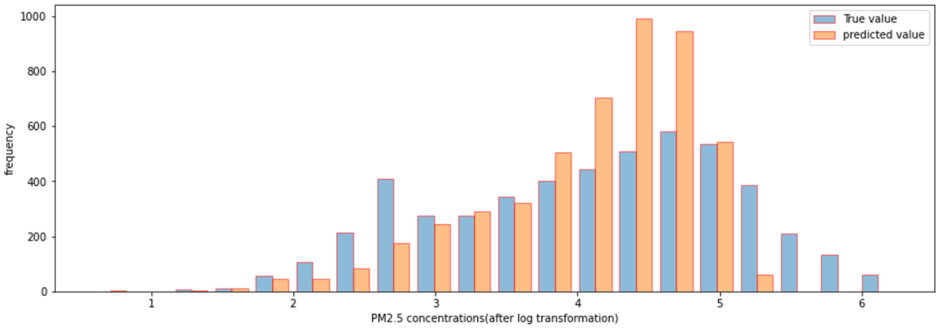

Figure 34

Score of LASSO regression (with CV):

```
(0.4753868019688896, 0.47850090471564144)
```

Effect of LASSO regression (with CV) (figure 35):

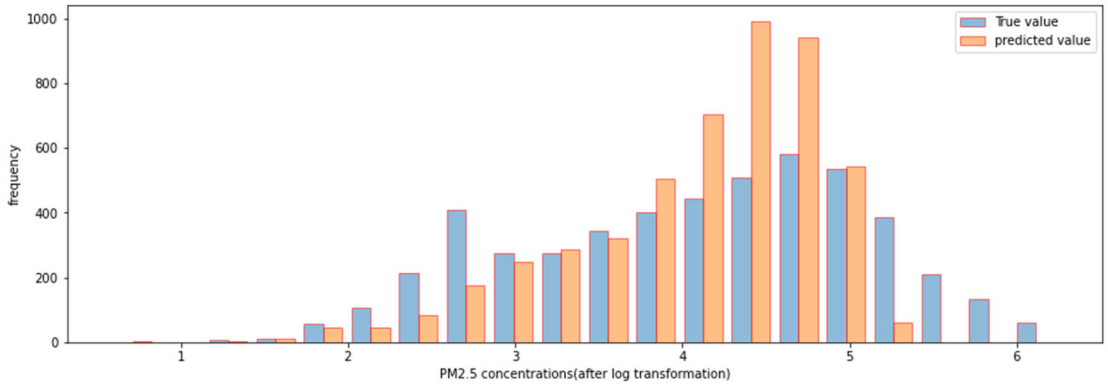

Figure 35

Score of XGBoosting regression:

```
(0.6369667410532611, 0.6076772229425884)
```

Effect of XGBoosting regression (figure 36):

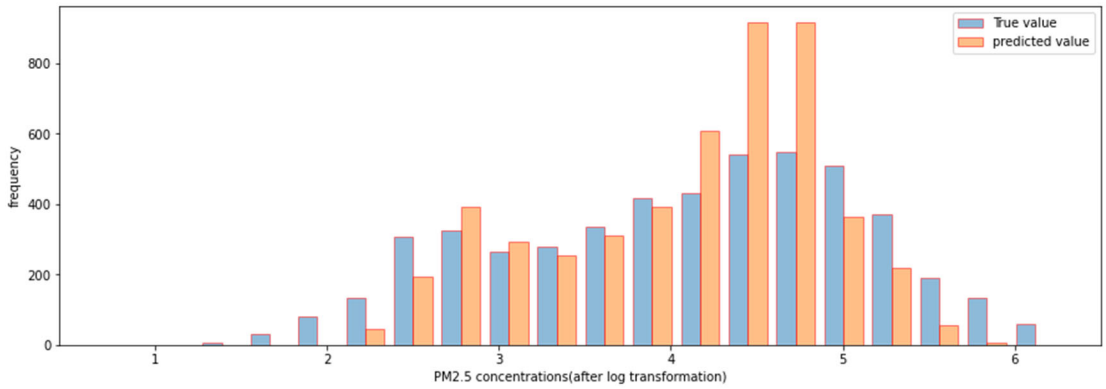

Figure 36

Score of Gradient Boosting Regression:

```
(0.6330496813566941, 0.6068923729545721)
```

Effect of Gradient Boosting Regression (figure 37):

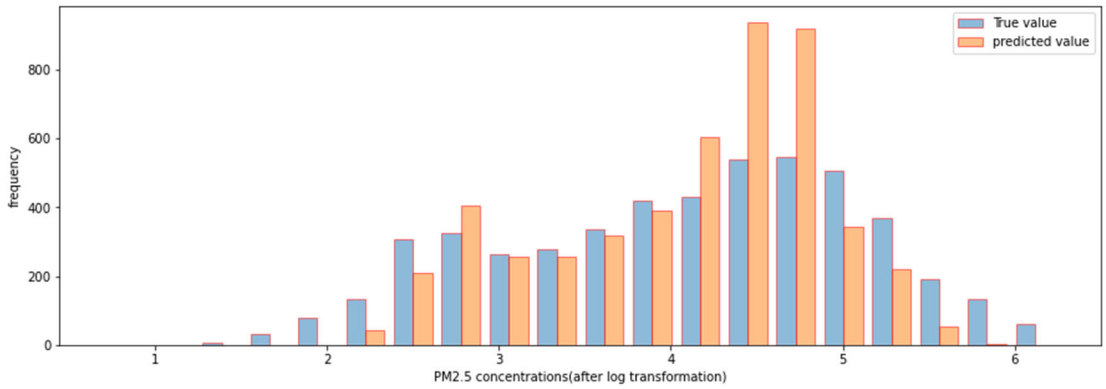

Figure 37

Score of Cat Regression:

```
(0.6574018959335886, 0.6138275445114958)
```

Effect of Cat Regression (figure 38):

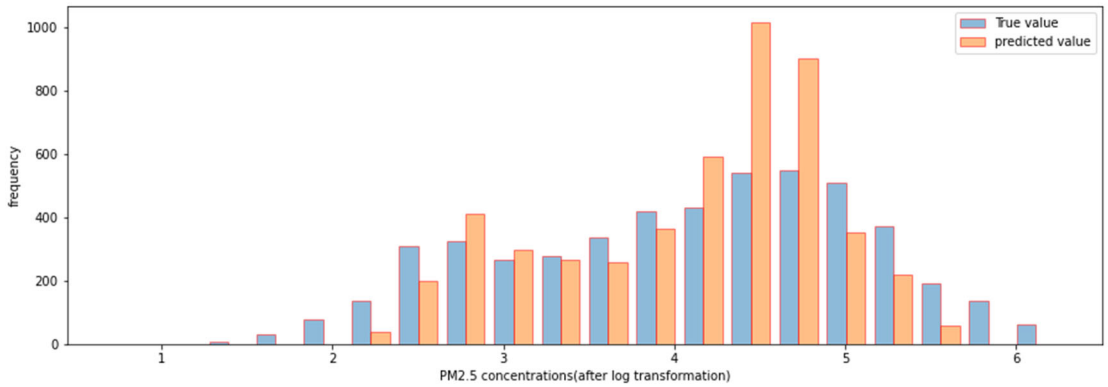

Figure 38

5.2 Fill the missing-value and do the same things as task 2

In the step 2 of task 2, we just delete the samples with missing data. Now, we select three methods to import the missing-value: average filling (denoted by filling method 1), kNN filling (denoted by filling method 2) and filling with the prediction value of an extreme forest regression model trained by the data without missing data (denoted by filling method 3).

Under each filling method, we do the same things as task 2 (except step 2), and we write the result here.

5.2.1 Fill the missing-value with filling method 1

Score of OLS:

```
(0.526641913319355, 0.5060910696235229)
```

Effect of OLS (figure 39):

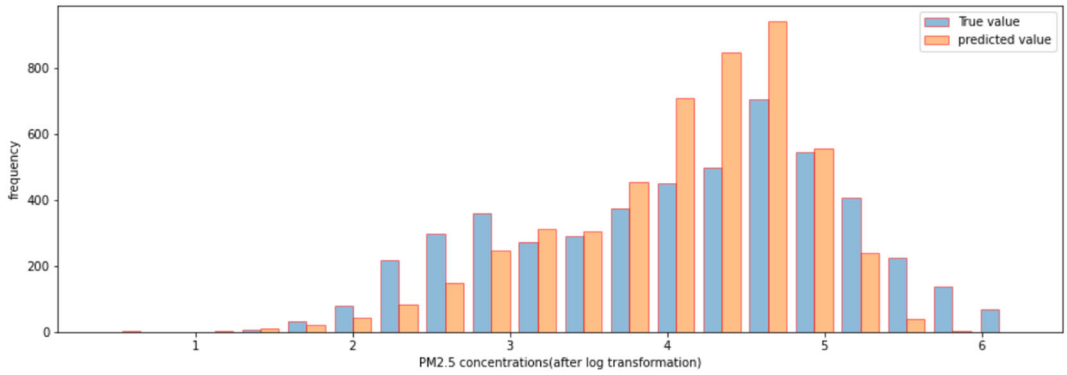

Figure 39

Score of ridge regression (with CV):

```
(0.5266417489341675, 0.5061287929987948)
```

Effect of ridge regression (with CV) (figure 40):

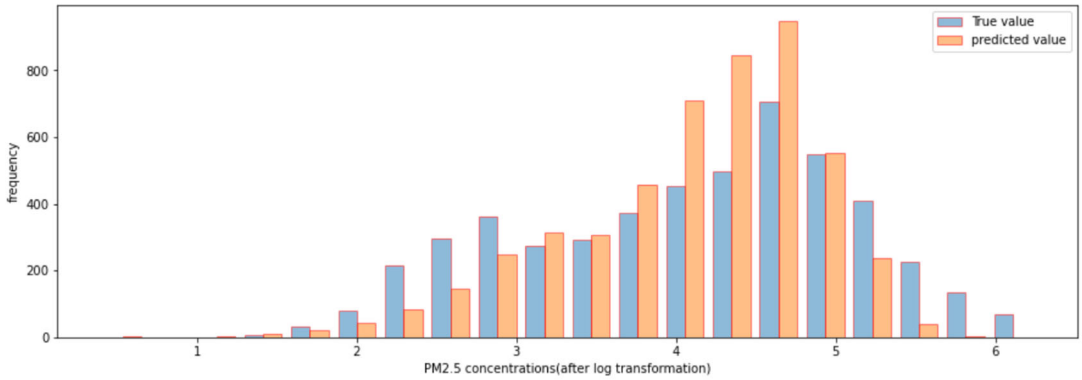

Figure 40

Score of LASSO regression (with CV):

```
(0.5266146279984567, 0.5065835599641144)
```

Effect of LASSO regression (with CV) (figure 41):

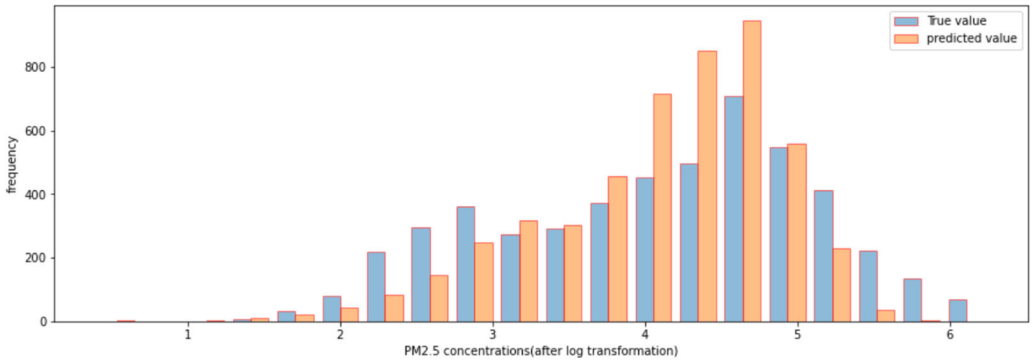

Figure 41

Score of XGBoosting regression:

```
(0.68403072148045, 0.612510915443903)
```

Effect of XGBoosting regression (figure 42):

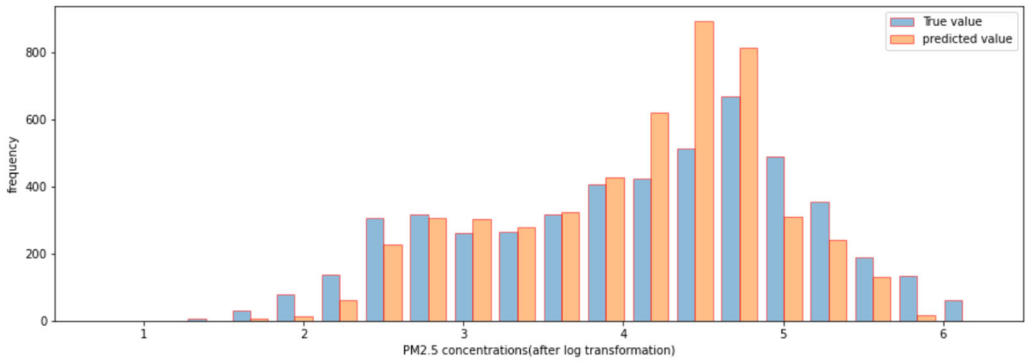

Figure 42

Score of Gradient Boosting Regression:

```
(0.6809976486966853, 0.6129264048989693)
```

Effect of Gradient Boosting Regression (figure 43):

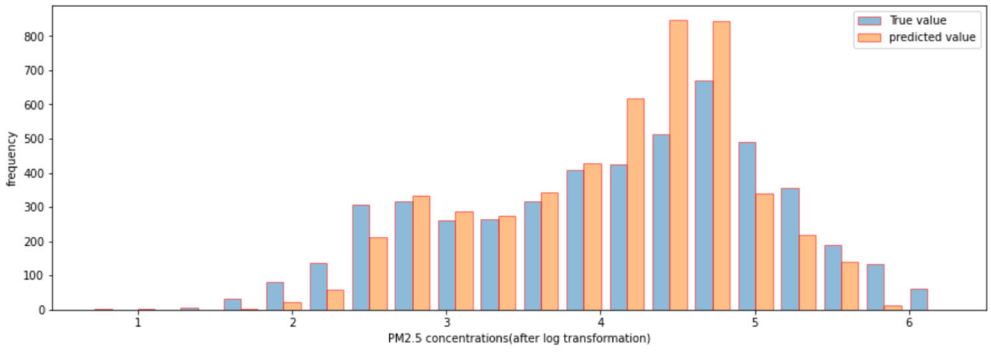

Figure 43

Score of Cat Regression:

```
(0.7198688946268785, 0.6204427224632725)
```

Effect of Cat Regression (figure 44):

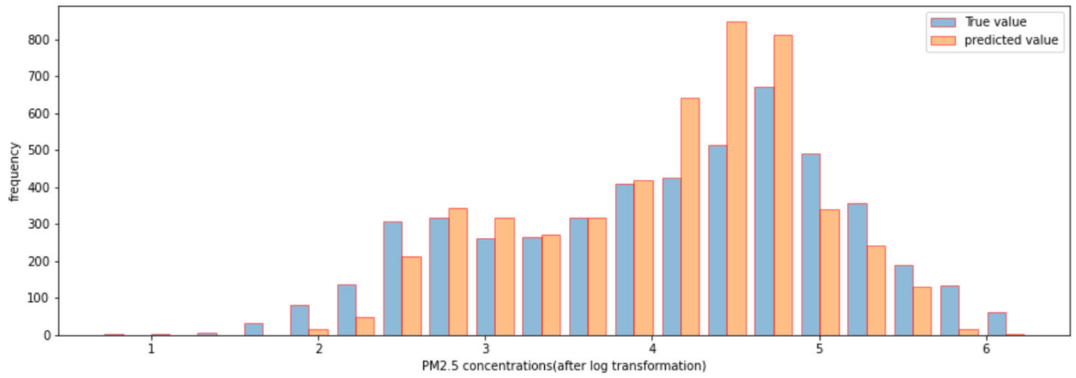

Figure 44

5.2.2 Fill the missing-value with filling method 2

Score of OLS:

```
(0.5574729570061818, 0.5424262245433004)
```

Effect of OLS (figure 45):

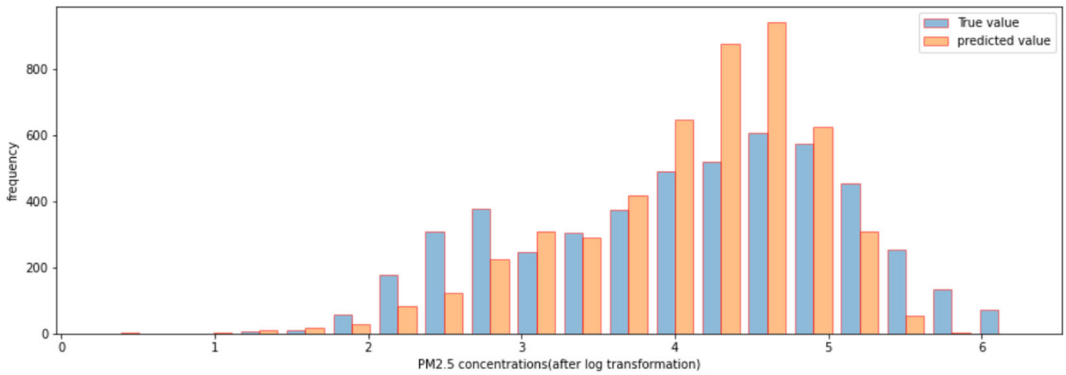

Figure 45

Score of ridge regression (with CV):

```
(0.5574728474676032, 0.5424541226673627)
```

Effect of ridge regression (with CV) (figure 46):

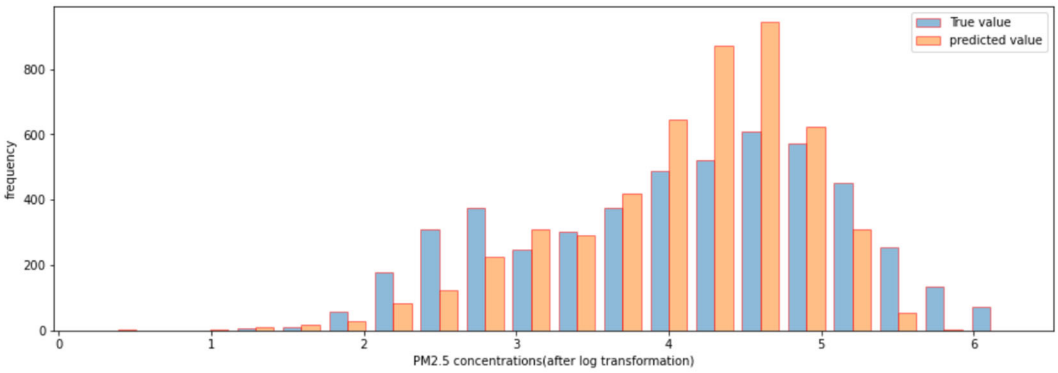

Figure 46

Score of LASSO regression (with CV):

```
(0.557430651311172, 0.5428594738413781)
```

Effect of LASSO regression (with CV) (figure 47):

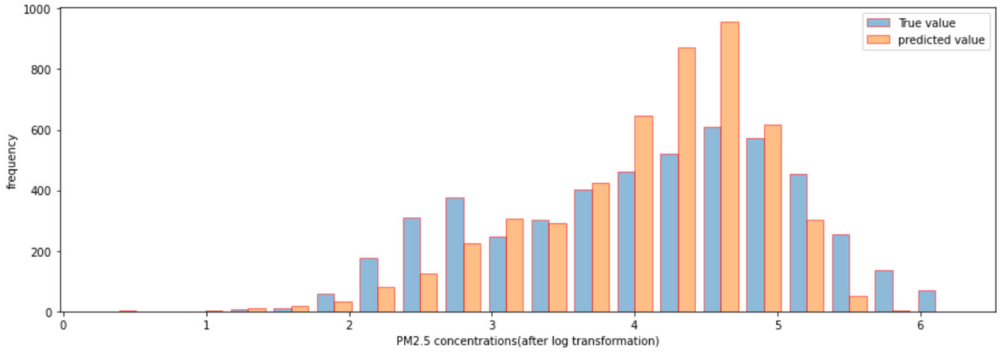

Figure 47

Score of XGBoosting regression:

```
(0.7097736552107224, 0.6457876544827008)
```

Effect of XGBoosting regression (figure 48):

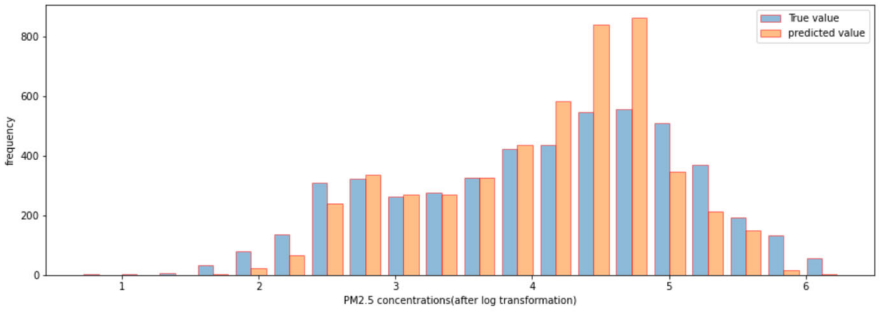

Figure 48

Score of Gradient Boosting Regression:

```
(0.711054958627863, 0.6468152887446903)
```

Effect of Gradient Boosting Regression (figure 49):

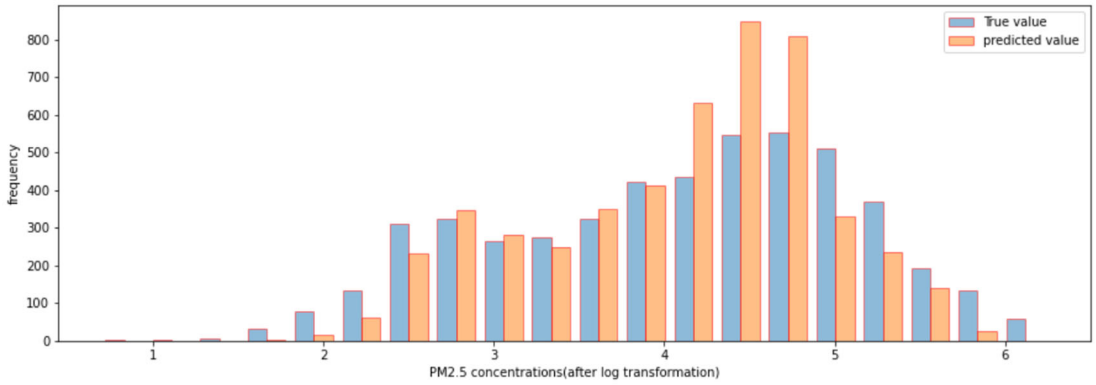

Figure 49

Score of Cat Regression:

```
(0.7425068974057469, 0.6542240774253631)
```

Effect of Cat Regression (figure 50):

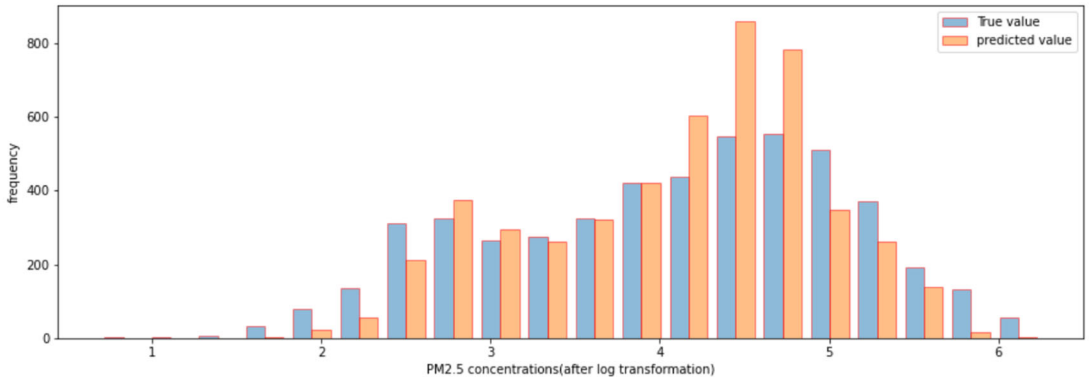

Figure 50

5.2.3 Fill the missing-value with filling method 3

Score of OLS:

```
(0.5639815885456352, 0.5458576364252119)
```

Effect of OLS (figure 51):

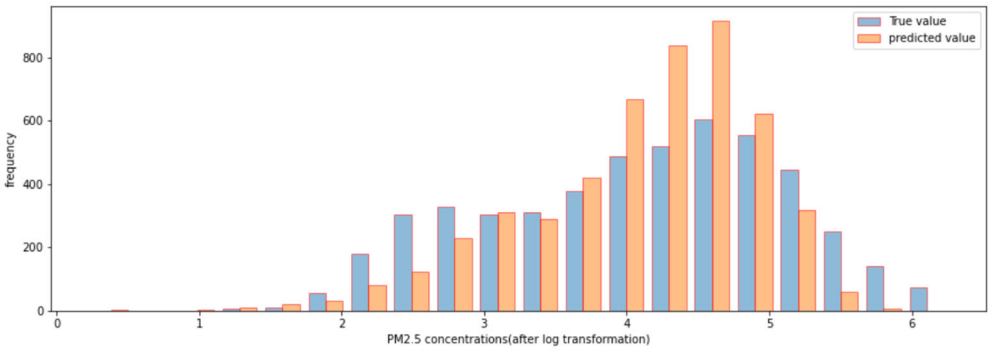

Figure 51

Score of ridge regression (with CV):

```
(0.5639814685754636, 0.5458873430006249)
```

Effect of ridge regression (with CV) (figure 52):

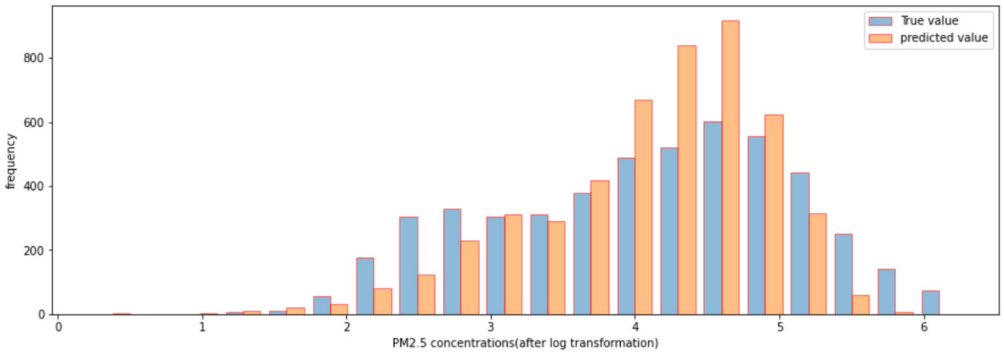

Figure 52

Score of LASSO regression (with CV):

```
(0.563954819328333, 0.54623147288161)
```

Effect of LASSO regression (with CV) (figure 53):

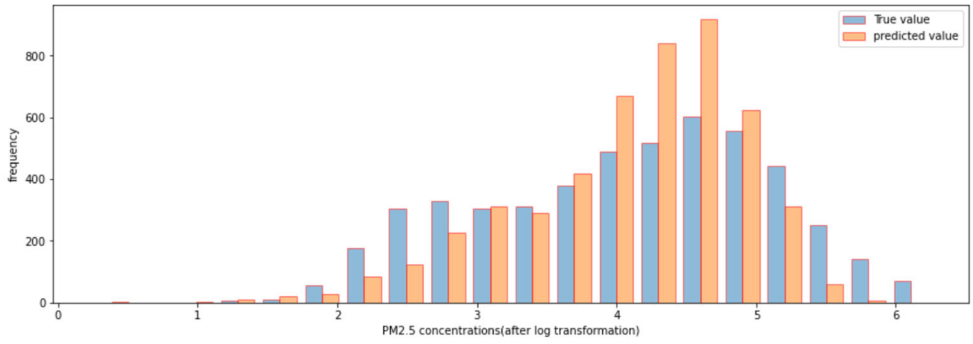

Figure 53

Score of XGBoosting regression:

```
(0.7163649233010376, 0.6537591814160111)
```

Effect of XGBoosting regression (figure 54):

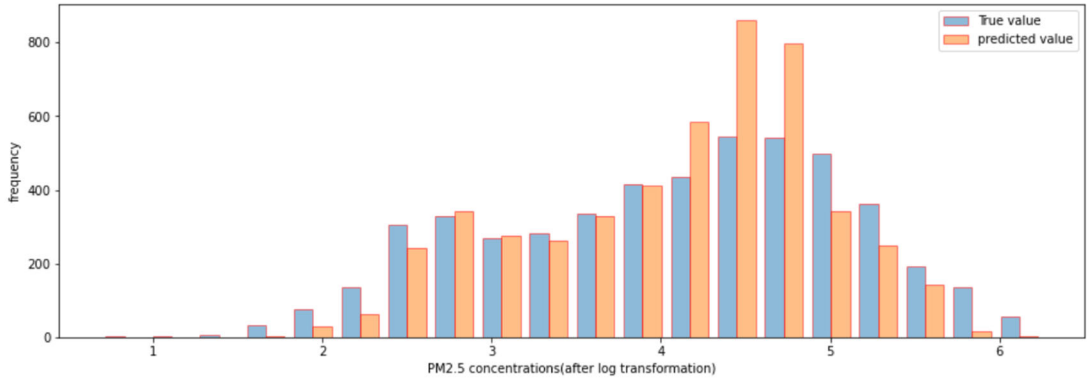

Figure 54

Score of Gradient Boosting Regression:

```
(0.7075973971772322, 0.6519002579277708)
```

Effect of Gradient Boosting Regression (figure 55):

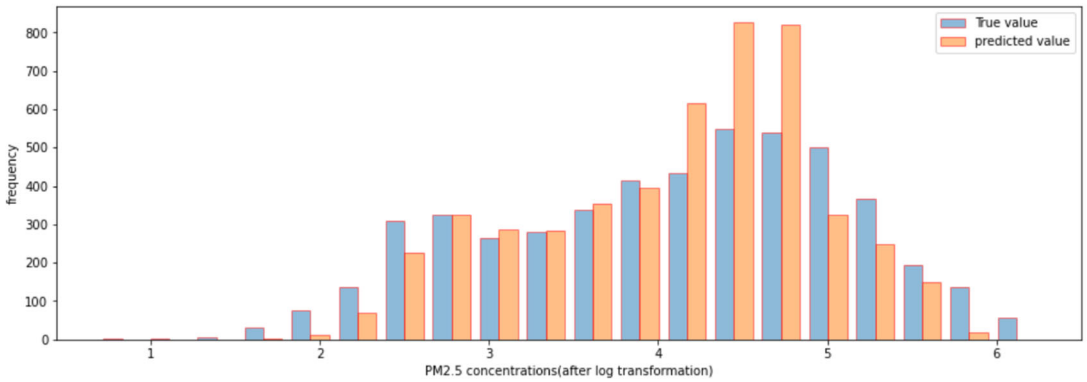

Figure 55

Score of Cat Regression:

```
(0.7474655074467347, 0.6604472983585946)
```

Effect of Cat Regression (figure 56):

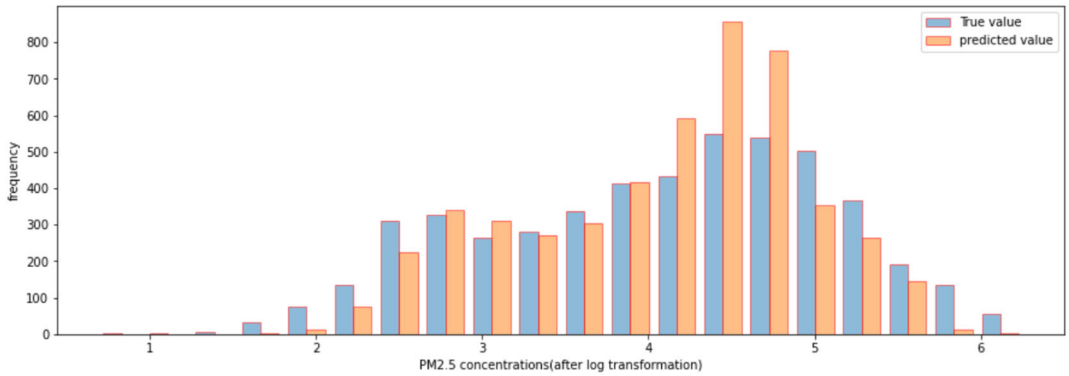

Figure 56

Summary of task 3: From above, we can know that the filling methods 1 and 2 fail to further improve the prediction accuracy of our models, but method 3 does. That is, we can fill the missing-value with the prediction value of an extreme forest regression model (with default parameters) trained by the data without missing data. After that, we select the Cat regression model (with n_estimators=1500, max_depth=6, learning_rate=0.039870, and keep the default values for other parameters). In this case, the model we trained has a score greater than 0.66 over the test set.

6. Task 4

For the preprocessing step, it is quite similar to that in Task2, which requires no more detailed explanations.

We will adopt several methods to do the dimension reduction, including PCA, kernel PCA, LLE, and ISOMAP. But mainly we will focus on the PCA method to analyze its performance and show the process of feature selecting.

6.1 Drop out missing values, Select the most important several features and Predict Pm2.5

First, after the processing and adding more features, we just delete the samples with missing data.

And then it comes to the step of feature selecting.

6.1.1 PCA method

Observing that the dimension of features now adds up to 21, we can first have a close look on the transformed 21 features, to get an overview about how many dimensions finally do we choose.

Importing the PCA module, after fitting and transforming, the 21 transformed features are ranked by their explained variances.

```
[2.20610184e+03 3.65205732e+02 2.67785958e+01 1.63856882e+01
 3.37383070e-01 2.52122975e-01 2.30881194e-01 2.19763784e-01
 2.19358465e-01 1.98349193e-01 1.73996834e-01 1.57052835e-01
 1.51090056e-01 1.30955526e-01 1.10145220e-01 8.54677073e-02
 7.00908832e-02 3.83970840e-02 5.00643751e-32 2.21465570e-32
 1.09289440e-32]
```

The corresponding explained variance ratio is as below.

```
[8.43038174e-01 1.39559456e-01 1.02331534e-02 6.26161512e-03
 1.28927324e-04 9.63460925e-05 8.82287736e-05 8.39803747e-05
 8.38254863e-05 7.57970182e-05 6.64910253e-05 6.00160574e-05
 5.77374457e-05 5.00432506e-05 4.20908152e-05 3.26605683e-05
 2.67844798e-05 1.46730341e-05 1.91315644e-35 8.46306941e-36
 4.17637881e-36]
```

The difference in level of magnitude starts significantly at n = 4, n = 15, and n = 18, so we shall later check whether they the ideal thresholds for PCA method.

The results can also be shown in a cumulating diagram of explained variance ratio with respect to the number of components more clearly. As seen below, the curve becomes nearly horizontal after the number reaches 4, so it gives us some reason to set the number of component to be 4.

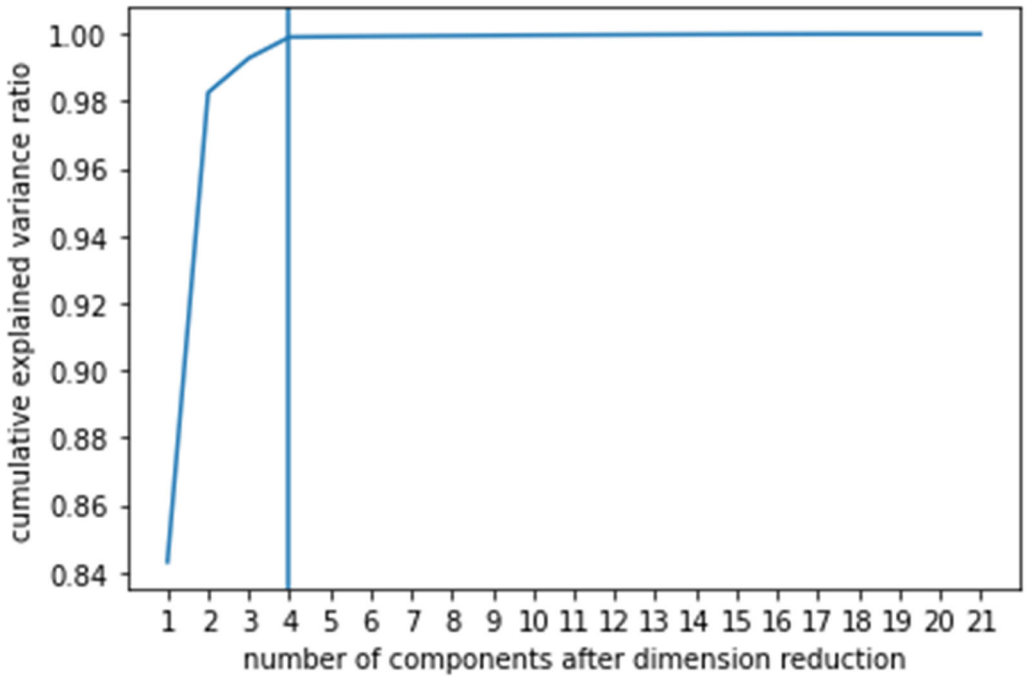

Figure 57

6.1.1.1 PCA, Choosing 18 transformed features having greatest variances

Here is the result of repeating the prediction task after choosing 18 features.

Score of OLS:

```
(0.5602751098902001, 0.5443339535991127)
```

Score of ridge regression (with CV):

```
(0.5602750668218053, 0.5443518894029715)
```

Score of ridge regression (with CV):

```
(0.5602749730898369, 0.5443658602840481)
```

Score of LASSO regression (with CV):

```
(0.3076518044344966, 0.3245999208448552)
```

Score of Cat Regression:

```
(0.7743612435020826, 0.6602303199193906)
```

6.1.1.2 PCA, Choosing 15 transformed features having greatest variances

Score of OLS:

```
(0.5109757268738448, 0.517566310710647)
```

Score of ridge regression:

```
(0.5109757240258599, 0.5175657163196686)
```

Score of ridge regression (with CV):

```
(0.5109751463855392, 0.5175572785353022)
```

Score of LASSO regression (with CV):

```
(0.3076518044344969, 0.32459992084485534)
```

Score of Cat Regression:

```
(0.7574256191748919, 0.6551184042915564)
```

6.1.1.3 PCA, Choosing 4 transformed features having greatest variances

Score of OLS:

```
(0.40623145151250195, 0.4086034028577682)
```

Score of ridge regression:

```
(0.40623145151189544, 0.4086034282872344)
```

Score of ridge regression (with CV):

```
(0.406231446473857, 0.40860571553586233)
```

Score of LASSO regression (with CV):

```
(0.3076518044351504, 0.3245999208453556)
```

Score of Cat Regression:

```
(0.5952200538985379, 0.5379217938233931)
```

6.1.2 Kernel PCA method

Score of OLS:

```
(0.1993925613619697, 0.21435924009799556)
```

Score of ridge regression:

```
(0.19939240860765417, 0.21432177009511344)
```

The performance of Kernel PCA is poorer than expected, since it also takes much time to run through all the boosting methods, we choose to stop now and try other ways of reduction of dimensions.

6.1.3 LLE

Score of OLS:

```
(0.36494450326681005, 0.4015315865556618)
```

Score of ridge regression:

```
(0.25777287430767926, 0.27967250052542825)
```

Score of ridge regression (with CV):

```
(0.3649444090216405, 0.40152418913692045)
```

Also, it is not better than PCA.

6.1.4 ISOMAP

Score of OLS:

```
(0.3517601272016573, 0.35009835514708687)
```

Score of ridge regression:

```
(0.351760127201421, 0.35009831224225607)
```

Score of ridge regression (with CV):

```
(0.3517600956720014, 0.35008265439319086)
```

Score of LASSO regression (with CV):

```
(0.31662889771702163, 0.29889052872735233)
```

Score of Cat Regression:

```
0.482157
```

6.2 Fill the missing data with extreme forest, Select the most important several features and Predict Pm2.5

Seeing that the performance of nom-linear dimension reduction methods is not as good as PCA in **6.1** section, and the computation of non-linear methods is need of over 8GB of RAM storage, we will only focus on PCA method for practice.

6.2.1 PCA, Choosing 4 transformed features having greatest variances

Score of OLS:

```
(0.40915567233981776, 0.4086972102101427)
```

Score of ridge regression:

```
(0.40915567233926453, 0.4086972394502909)
```

Score of ridge regression (with CV):

```
(0.4091556677434046, 0.40869987060103763)
```

Score of LASSO regression (with CV):

```
(0.30802330134603695, 0.32310598545986746)
```

Score of Cat Regression:

```
(0.5938734404379324, 0.5409718131484955)
```

6.3 Consider all possible useful features and then adopt PCA for dimension reduction (Drop out missing values)

Idea: the idea comes from Task2 that we can add our new features to make the prediction more accurate. Since PCA can always leave out the abundant information, in order not to miss any feature that is useful (for instance, a new label indication whether it is in month January), we can put all possible features that might be useful into our consideration.

Our practice is to rearrange the essential features (including 'year', 'month', and 'cbwd') by means of one-hot coding, because any single part of them may have a huge impact on the final result. In this case, we have 16 more features for prediction.

Importing the PCA module, after fitting and transforming, the 33 transformed features are ranked by their explained variances. Observing the chart below, we may choose n = 16, n = 28 as the truncation points.

```
[2.17831186e+03 3.65137528e+02 2.66318889e+01 1.63894340e+01
 3.74040322e-01 3.35306564e-01 2.52171690e-01 2.30805060e-01
 2.20410619e-01 2.15813640e-01 2.05559626e-01 1.95282651e-01
 1.86889735e-01 1.73642407e-01 1.52566475e-01 1.33856222e-01
 8.85648158e-02 8.73007231e-02 8.65656640e-02 8.50413052e-02
 8.16352201e-02 7.99293501e-02 7.78200010e-02 7.50420561e-02
 7.49323587e-02 6.27236311e-02 5.66172960e-02 7.02394720e-03
 4.29596267e-28 1.07045558e-29 1.07045558e-29 1.07045558e-29
 1.07045558e-29]
```

6.3.1 PCA, Choosing 33(all) features

Score of OLS

```
(0.5860676715008221, 0.5613402613125829)
```

Score of ridge regression:

```
(0.5860603470939887, 0.561465610322686)
```

Score of LASSO regression (with CV):

```
(0.30690999930373475, 0.32366693174757133)
```

Score of Cat Regression:

```
(0.8299640958183192, 0.6811108264752297)
```

6.3.2 PCA, Choosing 28(<33) transformed features having greatest variances

Score of OLS:

```
(0.586062221405365, 0.5613400026572031)
```

Score of ridge regression:

```
(0.5860603470939887, 0.561465610322686)
```

Score of LASSO regression (with CV):

```
(0.30690999930373475, 0.32366693174757133)
```

Score of Cat Regression:

```
(0.8300868067597248, 0.6826371565663263)
```

6.3.3 PCA, Choosing 16(<33) transformed features having greatest variances

Score of OLS:

```
(0.4912200488707259, 0.49599941182848917)
```

Score of ridge regression:

```
(0.49122004700657385, 0.4959989058672647)
```

Score of LASSO regression (with CV):

```
(0.30690999930403484, 0.32366693174779926)
```

Score of Cat Regression:

```
(0.7590528162727418, 0.6238527538059572)
```

Summary of task 4: In this task, we choose PCA, kernel PCA, LLE, and ISOMAP to do the feature selection and dimension reduction. But mainly we focused on the PCA

method to analyze its performance since non-linear methods have poor performance and take heavy computation.

and best score (R^2) over test set is nearly 0.682 shown in **section 6.3.2**, where we added more features (not existing previously) to be selected by PCA.

Although PCA can reduce the space complexity and curb the trend of over-fitting, the final R^2 does not go up significantly as we set fewer and fewer features (for example, only pick 4 features).

One ideal trade-off in **6.3.3** may be picking 28 features, which is less than the original feature size 33, and meanwhile preserving good R^2 result, i.e., 0.682.

7. Task 5

In Task 5, we partition the PM2.5 time series into three states: low PM state when PM2.5 ≤ 35$\mu g/m^3$; medium PM state when PM2.5 > 35 $\mu g/m^3$ and PM2.5 ≤ 150 $\mu g/m^3$; and high PM when PM2.5 > 150 $\mu g/m^3$. Then we treat the data set as a classification problem.

7.1 Delete the missing-values

Step 1: Introduce four features to deal with the feature 'cbwd' in the original data set. We call them 'cvornot', 'NEornot', 'NWornot' and 'SEornot'. If the word in ‘cbwd’ is ‘cv’, the corresponding 'cvornot' is 1, and other three variables are all 0. And similar thing for other cases.

Introduce four features to deal with samples in different hours. They are 'nightornot', 'morningornot', 'afternoonornot' and 'eveningornot'. If the hour of a sample is between 0 to 5, we set the 'nightornot' of it is 1, and other three features are 0. If the hour of a sample is between 6 to 11, we set the 'morningornot' of it is 1, and other three features are 0. If the hour of a sample is between 12 to 17, we set the 'afternoonornot' of it is 1, and other three features are 0. If the hour of a sample is between 18 to 23, we set the 'eveningornot' of it is 1, and other three features are 0.

Step 2: Let the data in ‘DEWP’, ‘TEMP’, ‘PRES’, ‘Iws’, ‘Is’, ‘Ir’ be smoothed. We use the smoothed method in the question. We process these columns of numbers, and each value is replaced by the average of itself, the preceding value and the following value (boundary values are not processed).

Step 3: Delete the samples with PM2.5 concentration ‘NA’.

Step 4: Use LOF algorithm to identify outliers and remove them.

Step 5: Give each sample a label. The rule is given in the start of ‘Task 5’.

Step 6: Split data to covariate vector X and response y. In this part, we choose 'year', 'month', 'day', 'DEWP', 'TEMP', 'PRES', 'Iws', 'Is', 'Ir', 'cvornot', 'NEornot', 'NWornot', 'SEornot', 'nightornot', 'morningornot', 'afternoonornot', 'eveningornot' as useful features in our later models.

Step 7: Divide all data into training set and test set (As the requirement in the task 1).

Step 8: Train a cat boosting classification model with the training set, and do some predictions.

Remark: We tried some common classification methods and finally selected cat boosting because of good effect.

To get a pair of better parameters, we did the following works: computing some scores over test set with different value of a parameter and compare them, fine-tuning the values of some parameters and select parameters that giving higher scores (They are clearly present in our code).

Finally, the parameters we used are 'n_estimators' = 1105, 'max_depth'= 6 and 'learning_rate' = 0.044375. And we keep the default values for other parameters.

These results is:

```
(0.8571528433120129, 0.7636629119732999)
```

Step 9: In order to visually display the classification results, we draw the confusion matrices (after normalization). Note that the entry in the row i and column j of the confusion matrix means the percentage of the sample with a real label i that was predicted to be j.

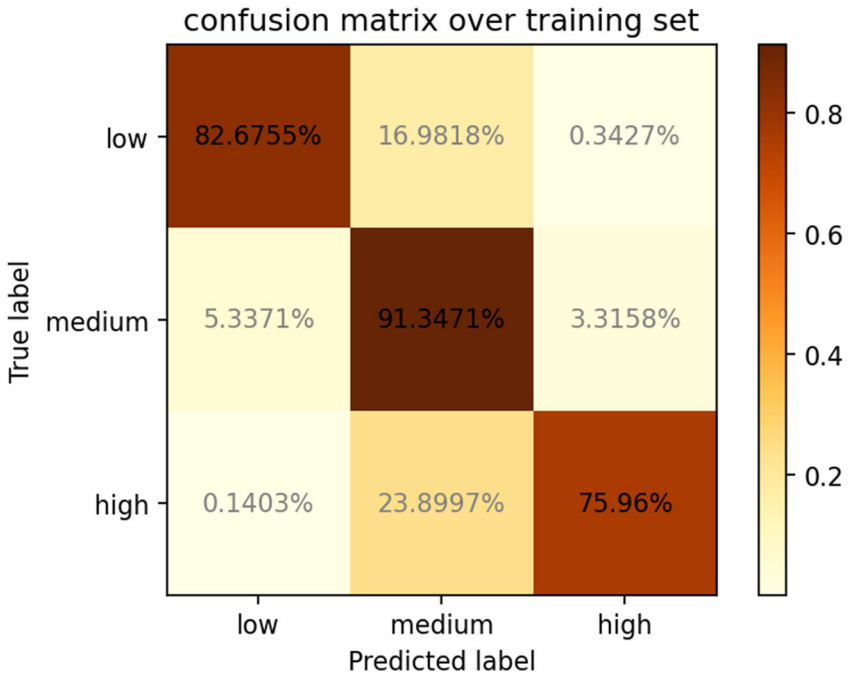

Figure 58

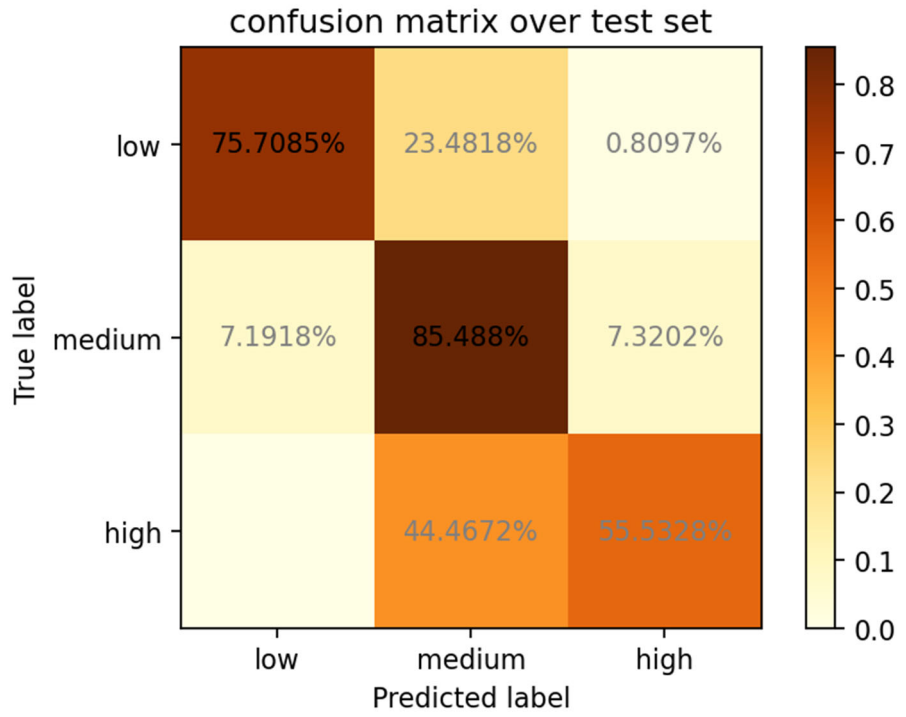

Figure 59

7.2 Fill the missing-values with extreme forest regression

Step 1: Introduce four features to deal with the feature 'cbwd' in the original data set. We call them 'cvornot', 'NEornot', 'NWornot' and 'SEornot'. If the word in 'cbwd' is 'cv', the corresponding 'cvornot' is 1, and other three variables are all 0. And similar thing for other cases.

Introduce four features to deal with samples in different hours. They are 'nightornot', 'morningornot', 'afternoonornot' and 'eveningornot'. If the hour of a sample is between 0 to 5, we set the 'nightornot' of it is 1, and other three features are 0. If the hour of a sample is between 6 to 11, we set the 'morningornot' of it is 1, and other three features are 0. If the hour of a sample is between 12 to 17, we set the 'afternoonornot' of it is 1, and other three features are 0. If the hour of a sample is between 18 to 23, we set the 'eveningornot' of it is 1, and other three features are 0.

Step 2: Let the data in 'DEWP', 'TEMP', 'PRES', 'Iws', 'Is', 'Ir' be smoothed. We use the smoothed method in the question. We process these columns of numbers, and each value is replaced by the average of itself, the preceding value and the following value

(boundary values are not processed).

Step 3: Fill the missing data with extreme forest regression (with default parameters) trained by the non-missing data set.

Step 4: Use LOF algorithm to identify outliers and remove them.

Step 5: Give each sample a label. The rule is given in the start of 'Task 5'.

Step 6: Split data to covariate vector X and response y. In this part, we choose 'year', 'month', 'day', 'DEWP', 'TEMP', 'PRES', 'Iws', 'Is', 'Ir', 'cvornot', 'NEornot', 'NWornot', 'SEornot', 'nightornot', 'morningornot', 'afternoonornot', 'eveningornot' as useful features in our later models.

Step 7: Divide all data into training set and test set (As the requirement in the task 1).

Step 8: Train a cat boosting classification model with the training set, and do some predictions.

Remark: We tried some common classification methods and finally selected cat boosting because of good effect.

To get a pair of better parameters, we did the following works: computing some scores over test set with different value of a parameter and compare them, fine-tuning the values of some parameters and select parameters that giving higher scores (They are clearly present in our code).

Finally, the parameters we used are 'n_estimators' = 1105, 'max_depth'= 6 and 'learning_rate' = 0.043450. And we keep the default values for other parameters.

These results is:

```
(0.8570906311276546, 0.7704225352112676)
```

Step 9: In order to visually display the classification results, we draw the confusion matrices (after normalization). Note that the entry in the row i and column j of the confusion matrix means the percentage of the sample with a real label i that was predicted to be j.

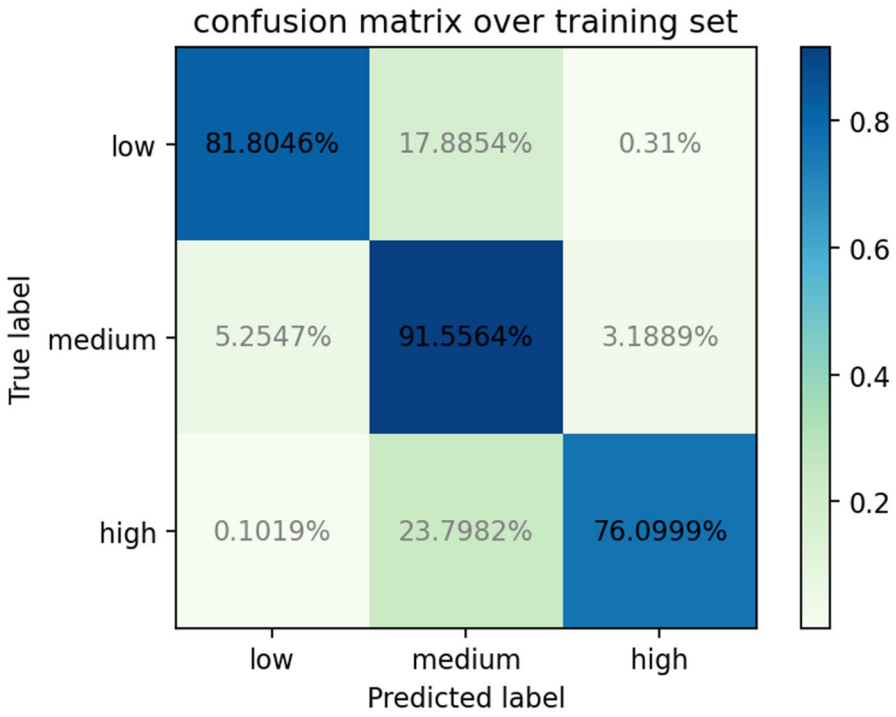

Figure 60

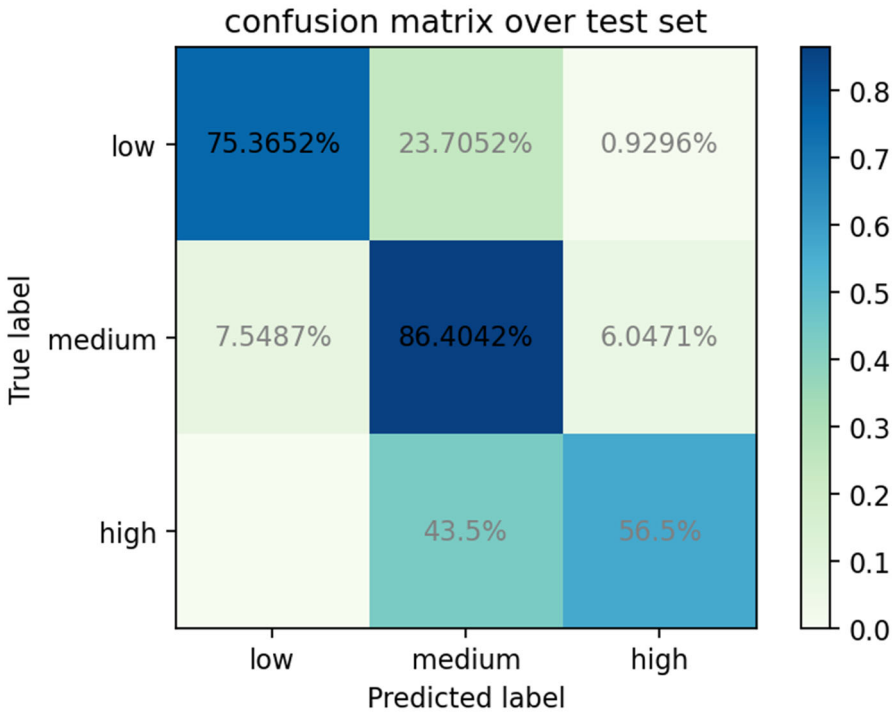

Figure 61

7.3 Reclassify training set and test set

We have noticed that the method of dividing the data into test set and training set provided in the question has great limitations: all samples of the test set are taken from Thursday, which lacks generality (generally speaking, there is a significant difference in PM2.5 concentration between weekends and non-weekends).

Thus, we want to add some samples in weekends to the test set. At the same time, in order to avoid the large number of samples in the test set, we adopt the following methods: if a sample is on Thursday and the value of hours is even, or it is on Sunday and the value of hours is odd, we divide it into the test set. And other samples are divided into the training set.

Other procedures are similar with **7.2**.

The model in this part is XG-boosting classification model trained by training set with parameters 'n_estimators'=1027, 'max_depth'=6, 'learning_rate'=0.17438425621293 (other value of parameters are default).

The scores are:

```
(0.9988038674950992, 0.8744457879887142)
```

The confusion matrices are:

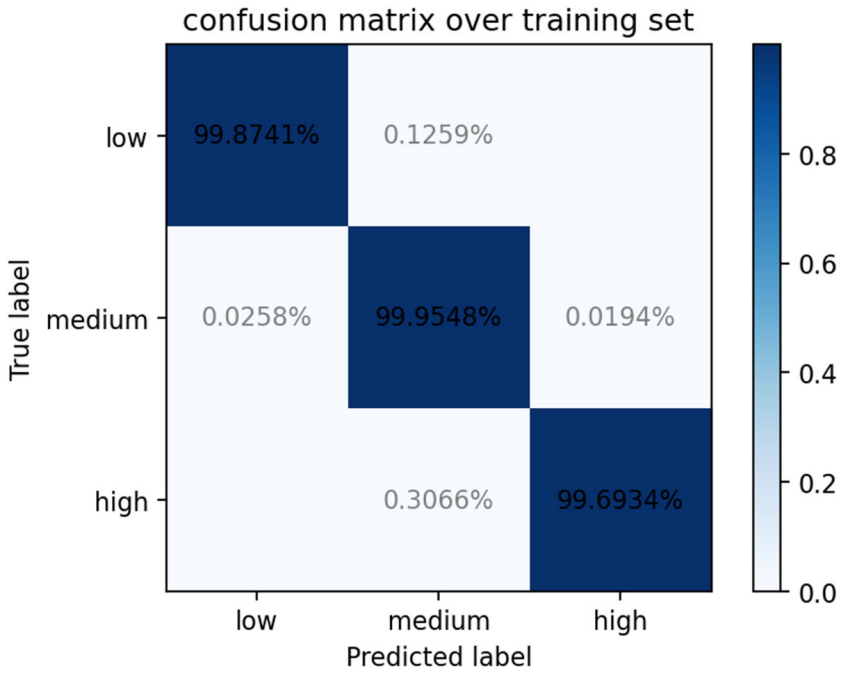

Figure 62

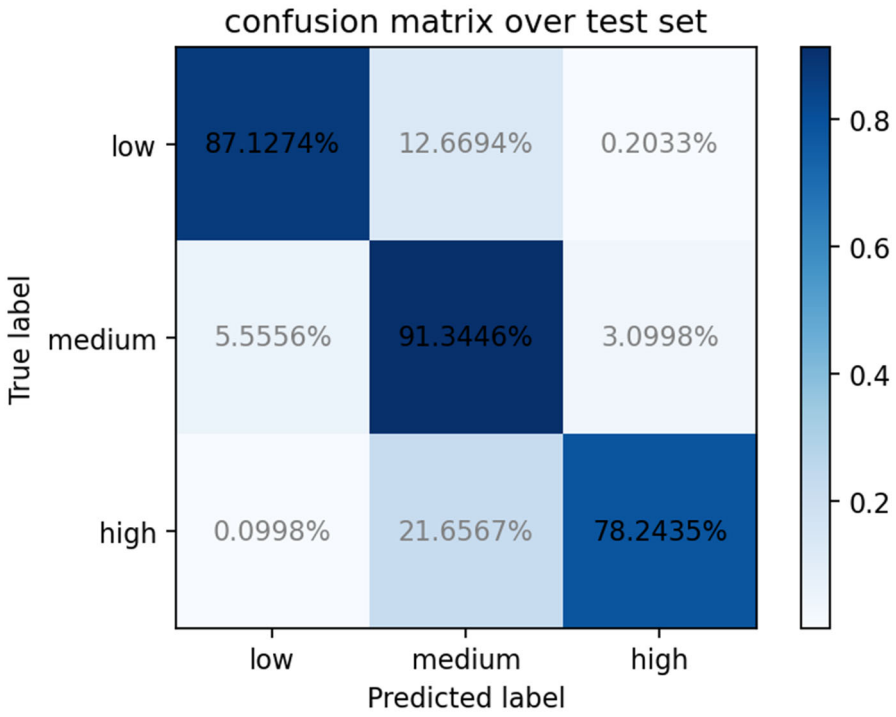

Figure 63

Summary of task 5: In this task, we choose cat boosting classification model to do prediction and the score over test set can be larger than 0.77. And if we change the method of dividing data set into training set and test set, the score over test set can be larger than 0.87!

Besides, from the confusion matrices, we can see that the effects of classifications are great. Because most of the samples were labeled correctly. And the proportion of high (low) concentration samples labeled as low (high) concentration in our model is extremely low. Especially, the model in 7.3 has very high scores both over train set and test set. And the confusion matrix over training set shows that in each true label, the percentage of samples successfully labeled by the model was greater than 99%. For the test set, our model also has a surprising performance!

8. Conclusions and probable improvement

8.1 Conclusion

We preprocessed the given data scientifically, did regression, assembled regression, feature selection (reduction of dimension) and classification on it, and finally achieved some effect both on the train and the test data sets.

And especially it is worth mentioning our classification method, which achieved a high accuracy according to confusion matrices.

8.2 Probable improvement

Firstly, our analysis only relies on the dependence between PM2.5 and the meteorological variables without considering the atmospheric chemical reaction that generates PM2.5 secondarily. Adding the chemical reaction into consideration may contribute to a better result.

Another prospect of elaborating the model prediction may lie in the refinement of time. Since for each month, the actual model governing Pm2.5 may vary so significantly according to unknown factors (for example, specific climate changes) such that it is not a wise way to put all things together. Instead, it might be better if the model is built basing on each month separately depending on concrete situations.