# HW3_309706045_傅可爲_Time Series Regression

```
import numpy as np
import pandas as pd
#讀取資料集
df = pd.read_csv("D:/新竹_2019.csv", engine="python")
df.columns = df.columns.str.strip()
df
```

| | 測站 | 日期 | 測項 | 00 | 01 | 02 | 03 | 04 | 05 | 06 | ... | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ---------- | ---------- | ---------- | -------- | -------- | -------- | -------- | -------- | -------- | -------- | ... | -------- | -------- | -------- | -------- | -------- | -------- | -------- | -------- | -------- | -------- |
| 1 | 新竹 | 2019/01/01 00:00:00 | AMB_TEMP | 17.5 | 17.4 | 17.3 | 17.1 | 16.8 | 16.9 | 16.8 | ... | 20 | 19.5 | 18.7 | 18.1 | 17.9 | 17.6 | 17.6 | 17.5 | 17.7 | 17.5 |
| 2 | 新竹 | 2019/01/01 00:00:00 | CH4 | 1.79 | 1.78 | 1.8 | 1.8 | 1.8 | 1.8 | 1.8 | ... | 1.81 | 1.81 | 1.81 | 1.8 | 1.8 | 1.81 | 1.81 | 1.8 | 1.8 | 1.79 |
| 3 | 新竹 | 2019/01/01 00:00:00 | CO | 0.19 | 0.21 | 0.22 | 0.22 | 0.21 | 0.21 | 0.23 | ... | 0.29 | 0.3 | 0.31 | 0.31 | 0.31 | 0.3 | 0.29 | 0.27 | 0.26 | 0.24 |
| 4 | 新竹 | 2019/01/01 00:00:00 | NMHC | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | 0.04 | ... | 0.06 | 0.06 | 0.07 | 0.07 | 0.07 | 0.06 | 0.05 | 0.04 | 0.04 | 0.04 |
| 5 | 新 | 2019/01/01 | NO | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | | 1.1 | 1.2 | 0.7 | 0.6 | 0.5 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 |

```
#將空白值的那一行 drop 掉，並選取 10~12 月的資料，並將測站跟日期去除
df = df.drop([0])
df = df[df["日期"].str.contains("2019/10|2019/11|2019/12")]
df = df.drop(columns=['測站', '日期'])
df
```

| | 測項 | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | ... | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4825 | AMB_TEMP | 24.7 | 25.1 | 25.4 | 25.5 | 25.3 | 25.1 | 24.6 | 24.6 | 25.1 | ... | 32.4 | 31.9 | 30.4 | 29.1 | 28.7 | 28.3 | 28 | 27.4 | 27 | 26.5 |
| 4826 | CH4 | 1.66 | 1.66 | 1.7 | 1.71 | 1.72 | 1.71 | 1.75 | 1.76 | 1.73 | ... | 1.69 | 1.72 | 1.74 | 1.74 | 1.78 | 1.82 | 1.82 | 1.83 | 1.93 | 1.96 |
| 4827 | CO | 0.05 | 0.13 | 0.15 | 0.17 | 0.16 | 0.16 | 0.22 | 0.42 | 0.36 | ... | 0.27 | 0.32 | 0.36 | 0.39 | 0.49 | 0.57 | 0.58 | 0.6 | 0.69 | 0.49 |
| 4828 | NMHC | 0 | 0 | 0 | 0.02 | 0.02 | 0.01 | 0.03 | 0.08 | 0.08 | ... | 0.07 | 0.1 | 0.08 | 0.08 | 0.13 | 0.21 | 0.22 | 0.21 | 0.22 | 0.17 |
| 4829 | NO | 0 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 1.2 | 5.3 | 5.6 | ... | 1.6 | 1.1 | 0.8 | 0.6 | 0.6 | 0.7 | 0.6 | 2 | 5.1 | 3.8 |
| 4830 | NO2 | 1.3 | 1.1 | 1.3 | 1.8 | 2.1 | 2.3 | 6.1 | 9.9 | 7.4 | ... | 5.9 | 8.3 | 10.1 | 9.6 | 14.9 | 19.5 | 22.2 | 26.7 | 24 | 15.7 |
| 4831 | NOx | 1.2 | 1.2 | 1.7 | 2.1 | 2.4 | 2.7 | 7.3 | 15.2 | 12.9 | ... | 7.4 | 9.4 | 10.8 | 10.3 | 15.5 | 20.2 | 22.8 | 28.6 | 29.1 | 19.5 |
| 4832 | O3 | 16.7 | 17.3 | 21.9 | 21.5 | 18.8 | 17.5 | 11.8 | 8.2 | 10.4 | ... | 41 | 54.1 | 57.7 | 49.3 | 39.1 | 30.8 | 23.3 | 13.1 | 7.5 | 5.5 |
| 4833 | PM10 | 18 | 18 | 29 | 29 | 27 | 19 | 23 | 13 | 10 | ... | 17 | 23 | 47 | 53 | 47 | 35 | 33 | 21 | 22 | 14 |
| 4834 | PM2.5 | 2 | 4 | 6 | 9 | 10 | 7 | 4 | 6 | 7 | ... | 3 | 8 | 19 | 17 | 19 | 19 | 19 | 14 | 17 | 8 |
| 4835 | RAINFALL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4836 | RH | 89 | 87 | 85 | 84 | 85 | 86 | 88 | 90 | 85 | ... | 51 | 56 | 64 | 66 | 66 | 68 | 70 | 72 | 74 | 74 |
| 4837 | SO2 | 1.1 | 1.3 | 1.4 | 1.5 | 1.5 | 1.6 | 1.8 | 2 | 1.9 | ... | 2 | 2.5 | 3.1 | 2.8 | 3.6 | 3.5 | 4.4 | 3.1 | 2.7 | 2.5 |
| 4838 | THC | 1.65 | 1.65 | 1.7 | 1.73 | 1.74 | 1.72 | 1.78 | 1.84 | 1.81 | ... | 1.76 | 1.82 | 1.82 | 1.82 | 1.91 | 2.03 | 2.04 | 2.04 | 2.15 | 2.13 |
| 4839 | WD_HR | 292 | 286 | 291 | 286 | 268 | 277 | 247 | 203 | 193 | ... | 241 | 243 | 249 | 248 | 251 | 245 | 128 | 186 | 176 | 178 |
| 4840 | WIND_DIREC | 283 | 289 | 276 | 289 | 250 | 262 | 239 | 183 | 187 | ... | 244 | 250 | 247 | 250 | 250 | 240 | 65 | 89 | 178 | 176 |
| 4841 | WIND_SPEED | 2.1 | 1.4 | 2 | 2.5 | 1.8 | 2.4 | 2 | 1.7 | 2.3 | ... | 3.3 | 3.6 | 4.9 | 4.4 | 2.8 | 1.7 | 1.2 | 0.5 | 0.9 | 1.1 |
| 4842 | WS_HR | 0.9 | 1.1 | 1.8 | 2.2 | 1.8 | 1.9 | 2 | 1 | 1.5 | ... | 2.1 | 2.2 | 3.8 | 4 | 2.8 | 1.5 | 0.5 | 0.4 | 0.7 | 0.9 |
| 4843 | AMB_TEMP | 26 | 25.7 | 25.3 | 25.2 | 25.2 | 25 | 25.2 | 26.4 | 28.6 | ... | 31 | 30.3 | 29.6 | 28.9 | 28.4 | 28.4 | 28.5 | 28.2 | 27.9 | 27.6 |
| 4844 | CH4 | 1.95 | 1.94 | 2.18 | 1.99 | 1.9 | 1.9 | 1.96 | 1.87 | 1.79 | ... | 1.92 | 1.91 | 1.87 | 1.87 | 1.88 | 1.91 | 1.97 | 2.04 | 2.1 | 2.08 |
| 4845 | CO | 0.4 | 0.31 | 0.35 | 0.3 | 0.25 | 0.27 | 0.48 | 0.64 | 0.46 | ... | 0.47 | 0.49 | 0.49 | 0.56 | 0.6 | 0.68 | 0.75 | 0.85 | 0.74 | 0.69 |
| 4846 | NMHC | 0.13 | 0.11 | 0.22 | 0.19 | 0.13 | 0.1 | 0.17 | 0.19 | 0.12 | ... | 0.07 | 0.07 | 0.09 | 0.12 | 0.11 | 0.13 | 0.17 | 0.21 | 0.25 | 0.2 |
| 4847 | NO | 3.8 | 0.9 | 6.9 | 10.3 | 5.8 | 1.2 | 13.8 | 15.6 | 7.3 | ... | 0.3 | 0.7 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 |
| 4848 | NO2 | 11.6 | 10.7 | 12.9 | 14.8 | 17.1 | 15.3 | 20 | 20.2 | 15.9 | ... | 7.4 | 10.7 | 7.8 | 8.5 | 9.6 | 13.4 | 17.2 | 27.8 | 21.4 | 19.9 |

```
#因為要做時序的預測，因此將直向的日期資料改為橫向
new_df = df.iloc[:18,:]
plot_size = int(len(df) / 18)
for i in range(plot_size-1):
    new_df = pd.merge(new_df, df.iloc[i * 18 + 18:i * 18 + 2 * 18,:],on='測項')
    #new_df = pd.concat(new_df, df.iloc[i * 18 + 18:i * 18 + 2 * 18,:],axis=1)
df = new_df
#df.iloc[0:18,20:30]
df
```

| | 測項 | 00_x | 01_x | 02_x | 03_x | 04_x | 05_x | 06_x | 07_x | 08_x | ... | 14_y | 15_y | 16_y | 17_y | 18_y | 19_y | 20_y | 21_y | 22_y | 23_y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | AMB_TEMP | 24.7 | 25.1 | 25.4 | 25.5 | 25.3 | 25.1 | 24.6 | 24.6 | 25.1 | ... | 16.3 | 15.9 | 15.4 | 15.3 | 15.3 | 15.1 | 15 | 15 | 15 | 15.2 |
| 1 | CH4 | 1.66 | 1.66 | 1.7 | 1.71 | 1.72 | 1.71 | 1.75 | 1.76 | 1.73 | ... | 1.79 | 1.79 | 1.78 | 1.78 | 1.77 | 1.75 | 1.74 | 1.74 | 1.74 | 1.74 |
| 2 | CO | 0.05 | 0.13 | 0.15 | 0.17 | 0.16 | 0.16 | 0.22 | 0.42 | 0.36 | ... | 0.4 | 0.39 | 0.4 | 0.43 | 0.43 | 0.36 | 0.31 | 0.3 | 0.29 | 0.29 |
| 3 | NMHC | 0 | 0 | 0 | 0.02 | 0.02 | 0.01 | 0.03 | 0.08 | 0.08 | ... | 0.1 | 0.11 | 0.11 | 0.12 | 0.11 | 0.09 | 0.07 | 0.07 | 0.07 | 0.06 |
| 4 | NO | 0 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 1.2 | 5.3 | 5.6 | ... | 1.6 | 1.4 | 1.1 | 1.1 | 0.8 | 0.8 | 0.3 | 0.8 | 0.6 | 0.3 |
| 5 | NO2 | 1.3 | 1.1 | 1.3 | 1.8 | 2.1 | 2.3 | 6.1 | 9.9 | 7.4 | ... | 9.8 | 10.8 | 11.3 | 12.8 | 12.7 | 10.8 | 8.8 | 8.6 | 7.8 | 6.9 |
| 6 | NOx | 1.2 | 1.2 | 1.7 | 2.1 | 2.4 | 2.7 | 7.3 | 15.2 | 12.9 | ... | 11.3 | 12.2 | 12.4 | 14 | 13.5 | 11.5 | 9.2 | 9.3 | 8.4 | 7.3 |
| 7 | O3 | 16.7 | 17.3 | 21.9 | 21.5 | 18.8 | 17.5 | 11.8 | 8.2 | 10.4 | ... | 33.4 | 31.4 | 31.4 | 30.6 | 31.2 | 33.3 | 34.1 | 34.1 | 34.1 | 33.7 |
| 8 | PM10 | 18 | 18 | 29 | 29 | 27 | 19 | 23 | 13 | 10 | ... | 22 | 27 | 32 | 26 | 24 | 22 | 24 | 27 | 28 | 30 |
| 9 | PM2.5 | 2 | 4 | 6 | 9 | 10 | 7 | 4 | 6 | 7 | ... | 14 | 18 | 20 | 17 | 18 | 12 | 13 | 10 | 12 | 11 |
| 10 | RAINFALL | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | RH | 89 | 87 | 85 | 84 | 85 | 86 | 88 | 90 | 85 | ... | 73 | 74 | 76 | 74 | 73 | 72 | 73 | 72 | 72 | 73 |

```
#將 columns 中的首尾空格去除
for rows in range(df.shape[0]):
    for cols in range(df.shape[1]):
        df.iloc[rows,cols] = df.iloc[rows,cols].strip()
#宣告異常值的符號為 symbol
symbol = ['x','#','*','A']
#尋找異常值，將異常值最右和最左的正常值相加除以 2 的 function
def rep(i,j):
    check_left = j
    check_right = j
    left_NF = 0
    right_NF = 0

    #尋找左邊的數值
    while(left_NF == 0):
        check_left -= 1
        l_check = 1
        for k in range(len(symbol)):
            if (symbol[k]) in df.iloc[i,check_left]:
                l_check = 0
```

```
            if(l_check == 1):
                left_NF = 1
                break

        #尋找右邊的數值
        while(right_NF == 0):
            check_right += 1
            r_check = 1
            for k in range(len(symbol)):
                if (symbol[k]) in df.iloc[i,check_right]:
                    r_check = 0
            if(r_check == 1):
                right_NF = 1
                break
        #異常值讓左右正常值平均取代
        new_df = str((float(df.iloc[i,check_left]) + float(df.iloc[rows,check_right]))/2.0)
        df.iloc[i,j] = new_df
# NR(無降雨)，用 0 取代，缺失值/無效值則以前後一小時平均值取代
for i in range(df.shape[0]):
        for j in range(df.shape[1]):
            if (df.iloc[i,j]=="NR"):
                df.iat[i,j]="0"
            for k in range(len(symbol)):
                if (symbol[k]) in df.iloc[i,j]:
                    if(j > 2):
                        rep(i,j)
#將測項的那一列 drop 掉
df = df.drop(columns=['測項'])
#將 10、11 月為訓練集，12 月為測試集
train_df = df.iloc[:,:1465]
test_df = df.iloc[:,1465:]
```

train_df

| | 00_x | 01_x | 02_x | 03_x | 04_x | 05_x | 06_x | 07_x | 08_x | 09_x | ... | 15_x | 16_x | 17_x | 18_x | 19_x | 20_x | 21_x | 22_x | 23_x | 00_y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 24.7 | 25.1 | 25.4 | 25.5 | 25.3 | 25.1 | 24.6 | 24.6 | 25.1 | 26.8 | ... | 25.1 | 23.6 | 22.4 | 22.1 | 21.7 | 21.2 | 21 | 20.9 | 20.8 | 20.5 |
| 1 | 1.66 | 1.66 | 1.7 | 1.71 | 1.72 | 1.71 | 1.75 | 1.76 | 1.73 | 1.73 | ... | 1.82 | 1.83 | 1.83 | 1.87 | 2 | 1.97 | 1.85 | 1.9 | 1.84 | 1.86 |
| 2 | 0.05 | 0.13 | 0.15 | 0.17 | 0.16 | 0.16 | 0.22 | 0.42 | 0.36 | 0.29 | ... | 0.4 | 0.41 | 0.42 | 0.56 | 0.69 | 0.71 | 0.48 | 0.5 | 0.34 | 0.28 |
| 3 | 0 | 0 | 0 | 0.02 | 0.02 | 0.01 | 0.03 | 0.08 | 0.08 | 0.08 | ... | 0.12 | 0.12 | 0.11 | 0.17 | 0.23 | 0.23 | 0.15 | 0.15 | 0.1 | 0.08 |
| 4 | 0 | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 | 1.2 | 5.3 | 5.6 | 4.8 | ... | 1.7 | 1 | 0.5 | 0.6 | 1.6 | 2.6 | 1 | 1.1 | 0.7 | 0.6 |
| 5 | 1.3 | 1.1 | 1.3 | 1.8 | 2.1 | 2.3 | 6.1 | 9.9 | 7.4 | 6 | ... | 22.9 | 20.1 | 13.7 | 24.5 | 33.4 | 31.8 | 22.3 | 22.7 | 16.5 | 13.7 |
| 6 | 1.2 | 1.2 | 1.7 | 2.1 | 2.4 | 2.7 | 7.3 | 15.2 | 12.9 | 10.8 | ... | 24.6 | 21.1 | 14.2 | 25.1 | 34.9 | 34.4 | 23.2 | 23.7 | 17.2 | 14.3 |
| 7 | 16.7 | 17.3 | 21.9 | 21.5 | 18.8 | 17.5 | 11.8 | 8.2 | 10.4 | 13.2 | ... | 52.1 | 48.1 | 51.8 | 37 | 21 | 15.4 | 23 | 16.4 | 20.8 | 21.5 |
| 8 | 18 | 18 | 29 | 29 | 27 | 19 | 23 | 13 | 10 | 13 | ... | 51 | 41 | 38 | 42 | 56 | 56 | 51 | 41 | 47 | 41 |
| 9 | 2 | 4 | 6 | 9 | 10 | 7 | 4 | 6 | 7 | 6 | ... | 35 | 17 | 20 | 22 | 34 | 36 | 32 | 27 | 22 | 18 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 89 | 87 | 85 | 84 | 85 | 86 | 88 | 90 | 85 | 75 | ... | 54 | 65 | 75 | 78 | 79 | 81 | 83 | 84 | 85 | 86 |

```
#導入需要使用的模組
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
from sklearn.metrics import mean_absolute_error
#記得要將 list 中的 string 都轉成 float 型態，才可以放進 model 去訓練
#訓練集 a-1(切割後的長度應為 1464-6=1458)
y1=[]
#只取第九行的 PM2.5 那一行，取第七個數開始以後的數放入 list(y1)
for i in range(1458):
    temp= train_df.iloc[9,i+6]
    y1.append(float(temp))
print(y1)
x1=[]
#只取第九行的 PM2.5 那一行，從第一個數每次取 6 小時為一單位切割並將之放入 list(x1)
for j in range(1458):
    temp=[]
    for k in range(6):
        temp.append(float(train_df.iloc[9,j+k]))
    x1.append(temp)
print(x1)
#print(len(x1))
#測試集 a-1(測試集切割後的長度應為 744-6=738)
y1_test=[]
#只取第九行的 PM2.5 那一行，取第七個數開始以後的數放入 list(y1_test)
for i in range(738):
    temp= train_df.iloc[9,i+6]
```

```
        y1_test.append(float(temp))
print(y1_test)
x1_test=[]
#只取第九行的 PM2.5 那一行，從第一個數每次取 6 小時為一單位切割並將之放入 list(x1_test)
for j in range(738):
        temp=[]
        for k in range(6):
                temp.append(float(train_df.iloc[9,j+k]))
        x1_test.append(temp)
print(x1_test)
#將未來第一個小時當預測目標，只取 PM2.5 的特徵(線性回歸)
#建立模型
reg = LinearRegression().fit(x1, y1)
#預測
y1_test_pred = reg.predict(x1_test)
print("未來第一個小時當預測目標\n 只取 PM2.5 的特徵\n 線性回歸的 MAE:")
#計算 MAE
mean_absolute_error(y1_test, y1_test_pred)
```

```
未來第一個小時當預測目標
只取PM2.5的特徵
線性回歸的MAE:

3.0924194644045326
```

```
#將未來第一個小時當預測目標，只取 PM2.5 的特徵(隨機森林)
#建立模型
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(x1, y1)
#預測
y1_test_pred = regr.predict(x1_test)
print("未來第一個小時當預測目標\n 只取 PM2.5 的特徵\n 隨機森林的 MAE:")
#計算 MAE
mean_absolute_error(y1_test, y1_test_pred)
```

```
未來第一個小時當預測目標
只取PM2.5的特徵
隨機森林的MAE:

3.3062048056392146
```

```
#訓練集 a-2(切割後的長度應為 1464-11=1453)
y2=[]
#只取第九行的 PM2.5 那一行，取第十二個數開始以後的數放入 list(y2)
for i in range(1453):
    temp= train_df.iloc[9,i+11]
    y2.append(float(temp))
print(y2)
x2=[]
#只取第九行的 PM2.5 那一行，從第一個數每次取 6 小時為一單位切割並將之放入 list(x2)
for j in range(1453):
    temp=[]
    for k in range(6):
        temp.append(float(train_df.iloc[9,j+k]))
    x2.append(temp)
print(x2)

#測試集 a-2(測試集切割後的長度應為 744-11=733)
y2_test=[]
#只取第九行的 PM2.5 那一行，取第十二個數開始以後的數放入 list(y2_test)
for i in range(733):
    temp= train_df.iloc[9,i+11]
    y2_test.append(float(temp))
print(y2_test)
x2_test=[]
#只取第九行的 PM2.5 那一行，從第一個數每次取 6 小時為一單位切割並將之放入 list(x2_test)
for j in range(733):
    temp=[]
    for k in range(6):
        temp.append(float(train_df.iloc[9,j+k]))
    x2_test.append(temp)
print(x2_test)
#將未來第六個小時當預測目標，只取 PM2.5 的特徵(線性回歸)
#建立模型
reg = LinearRegression().fit(x2, y2)
#預測
y2_test_pred = reg.predict(x2_test)
```

```
print("未來第六個小時當預測目標\n 只取 PM2.5 的特徵\n 線性回歸的 MAE:")
#計算 MAE
mean_absolute_error(y2_test, y2_test_pred)
```

```
未來第六個小時當預測目標
只取PM2.5的特徵
線性回歸的MAE:

5.175525364452278
```

```
#將未來第六個小時當預測目標，只取 PM2.5 的特徵(隨機森林)
#建立模型
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(x2, y2)
#預測
y2_test_pred = regr.predict(x2_test)
print("未來第六個小時當預測目標\n 只取 PM2.5 的特徵\n 隨機森林的 MAE:")
#計算 MAE
mean_absolute_error(y2_test, y2_test_pred)
```

```
未來第六個小時當預測目標
只取PM2.5的特徵
隨機森林的MAE:

5.08502835176121
```

```
#訓練集 b-1(切割後的長度應為 1464-6=1458)
yb1=[]
#只取第九行的 PM2.5 那一行，取第七個數開始以後的數放入 list(yb1)
for i in range(1458):
    temp= train_df.iloc[9,i+6]
    yb1.append(float(temp))
print(yb1)
xb1=[]
#取所有 18 種屬性*6，把每 108 個特徵放入 temp_ar 再將之放入 list(xb1)
for j in range(1458):
    temp_ar=[]
    for l in range(18):
        for k in range(6):
            temp=train_df.iloc[l,j+k]
            temp_ar.append(float(temp))
    xb1.append(temp_ar)
print(xb1)
```

```
#測試集 b-1(切割後的長度應為 744-6=738)
yb1_test=[]
#只取第九行的 PM2.5 那一行，取第七個數開始以後的數放入 list(yb1_test)
for i in range(738):
    temp= train_df.iloc[9,i+6]
    yb1_test.append(float(temp))
print(yb1_test)
xb1_test=[]
#取所有 18 種屬性*6，把每 108 個特徵放入 temp_ar 再將之放入 list(xb1_test)
for j in range(738):
    temp_ar=[]
    for l in range(18):
        for k in range(6):
            temp=train_df.iloc[l,j+k]
            temp_ar.append(float(temp))
    xb1_test.append(temp_ar)
print(xb1_test)

#將未來第一個小時當預測目標，取所有 18*6 個特徵(線性回歸)
#建立模型
reg = LinearRegression().fit(xb1, yb1)
#預測
yb1_test_pred = reg.predict(xb1_test)
print("未來第一個小時當預測目標\n 取所有 18 種屬性\n 線性回歸的 MAE:")
#計算 MAE
mean_absolute_error(yb1_test, yb1_test_pred)
```

```
未來第一個小時當預測目標
取所有18種屬性
線性回歸的MAE:

2.7582650584478
```

```
#將未來第一個小時當預測目標，取所有 18*6 個特徵(隨機森林)
#建立模型
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(xb1, yb1)
#預測
yb1_test_pred = regr.predict(xb1_test)
print("未來第一個小時當預測目標\nX 取所有 18 種屬性\n 隨機森林的 MAE:")
#計算 MAE
```

```
mean_absolute_error(yb1_test, yb1_test_pred)
```

```
未來第一個小時當預測目標
X取所有18種屬性
隨機森林的MAE:
3.3062048056392146
```

```
#訓練集 b-2(切割後的長度應為 1464-11=1453)
yb2=[]
#只取第九行的 PM2.5 那一行，取第十二個數開始以後的數放入 list(yb2)
for i in range(1453):
    temp= train_df.iloc[9,i+11]
    yb2.append(float(temp))
print(yb2)
xb2=[]
#取所有 18 種屬性*6，把每 108 個特徵放入 temp_ar 再將之放入 list(xb2)
for j in range(1453):
    temp_ar=[]
    for l in range(18):
        for k in range(6):
            temp=train_df.iloc[l,j+k]
            temp_ar.append(float(temp))
    xb2.append(temp_ar)
print(xb2)

#測試集 b-2(切割後的長度應為 744-11=733)
yb2_test=[]
#只取第九行的 PM2.5 那一行，取第十二個數開始以後的數放入 list(yb2_test)
for i in range(733):
    temp= train_df.iloc[9,i+11]
    yb2_test.append(float(temp))
print(yb2_test)
xb2_test=[]
#取所有 18 種屬性*6，把每 108 個特徵放入 temp_ar 再將之放入 list(xb2_test)
for j in range(733):
    temp_ar=[]
    for l in range(18):
        for k in range(6):
            temp=train_df.iloc[l,j+k]
            temp_ar.append(float(temp))
    xb2_test.append(temp_ar)
```

```
print(xb2_test)
#將未來第六個小時當預測目標，取所有 18*6 個特徵(線性回歸)
#建立模型
reg = LinearRegression().fit(xb2, yb2)
#預測
yb2_test_pred = reg.predict(xb2_test)
print("未來第六個小時當預測目標\n 取所有 18 種屬性\n 線性回歸的 MAE:")
#計算 MAE
mean_absolute_error(yb2_test, yb2_test_pred)
```

```
未來第六個小時當預測目標
取所有18種屬性
線性回歸的MAE:

4.708597810336621
```

```
#將未來第六個小時當預測目標，取所有 18*6 個特徵(隨機森林)
#建立模型
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(xb2, yb2)
#預測
yb2_test_pred = regr.predict(xb2_test)
print("未來第六個小時當預測目標\n 取所有 18 種屬性\n 隨機森林的 MAE:")
#計算 MAE
mean_absolute_error(yb2_test, yb2_test_pred)
```

```
未來第六個小時當預測目標
取所有18種屬性
隨機森林的MAE:

4.960170970688833
```