Time Series Regression

學號 30706013
姓名 王聖晴

目錄

CONTENTS

01 資料前處理 – 讀取資料與去掉空格

```python
data = pd.read_csv('新竹_2020.csv', encoding='Big5')

data = data[1:]  # 一開始第一行是無資料的row
# 去除資料中的空白
def rstrip(input_data):
    return [data.rstrip() for data in input_data]
# 移除columns 及 data中多餘的空格
data.columns = rstrip(data.columns)
for i in range(len(data)):
    data.iloc[i] = rstrip(data.iloc[i])
data
```

此份資料集有許多多餘的空白，因此在讀取時，針對欄位與值去掉多餘的空白，以利後續抓取資料。

	測站	日期	測項	00	01	02	03	04	05	06	...	14	15	16	17	18	19	20	21	22	23
1	新竹	2020/01/01 00:00:00	AMB_TEMP	15.2	15.2	15.3	15.3	15.3	15.4	15.5	...	18.1	18.2	17.9	17.3	16.7	16.4	16.2	16.1	16	15.8
2	新竹	2020/01/01 00:00:00	CH4	1.74	1.74	1.77	1.78	1.77	1.77	1.77	...	1.78	1.78	1.77	1.8	1.81	1.82	1.85	1.83	1.92	1.94
3	新竹	2020/01/01 00:00:00	CO	0.28	0.25	0.24	0.22	0.2	0.19	0.2	...	0.28	0.29	0.28	0.34	0.39	0.41	0.46	0.49	0.58	0.52
4	新竹	2020/01/01 00:00:00	NMHC	0.06	0.07	0.05	0.05	0.05	0.05	0.07	...	0.09	0.09	0.07	0.08	0.12	0.12	0.16	0.14	0.17	0.2
5	新竹	2020/01/01 00:00:00	NO	0.3	0.6	0.6	0.6	0.3	0.3	0.5	...	1.6	1.6	1.2	0.7	0.9	1.1	1.1	1.7	1.8	1.4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6584	新竹	2020/12/31 00:00:00	THC	2.01	2.02	2	2	1.99	2	1.98	...	2.03	2.07	2.07	2.1	2.1	2.07	2.07	2.05	2.04	2.07
6585	新竹	2020/12/31 00:00:00	WD_HR	54	55	54	53	58	52	52	...	54	50	52	45	47	42	42	47	45	44
6586	新竹	2020/12/31 00:00:00	WIND_DIREC	53	52	57	58	49	54	36	...	48	43	44	33	50	40	46	46	51	38
6587	新竹	2020/12/31 00:00:00	WIND_SPEED	4.7	4.6	4.7	4.9	4.1	5.3	5.5	...	4.5	4.4	4.2	3.8	3.7	4.7	4.5	4.4	3.9	3.9
6588	新竹	2020/12/31 00:00:00	WS_HR	3.7	3.6	3.6	3.5	3.5	3.3	3.8	...	3.7	3.1	3.3	3.1	2.9	3.3	3.1	2.9	2.8	2.6

01 資料前處理 – 取出10-12月資料

```
1  date = data['日期'] >= '2020/10/01'
2  data = pd.DataFrame(data[date])
3  data['日期'] = pd.to_datetime(data['日期'])
4  data
```

	測站	日期	測項	00	01	02	03	04	05	06	...	14	15	16	17	18	19	20	21	22	23
4933	新竹	2020-10-01	AMB_TEMP	23.7	23.8	23.8	23.9	23.9	23.8	24.1	...	29.9	29.6	28.7	27.5	26.4	25.7	25.5	25.3	24.9	24.5
4934	新竹	2020-10-01	CH4	1.97	1.95	1.96	1.96	1.95	1.96	1.97	...	1.97	1.98	1.97	2	2.03	2.04	2.05	2.02	2.1	2.14
4935	新竹	2020-10-01	CO	0.23	0.22	0.21	0.2	0.2	0.22	0.24	...	0.29	0.3	0.33	0.38	0.46	0.5	0.45	0.39	0.46	0.45
4936	新竹	2020-10-01	NMHC	0.06	0.05	0.03	0.03	0.03	0.04	0.04	...	0.06	0.07	0.09	0.11	0.13	0.15	0.1	0.07	0.12	0.18
4937	新竹	2020-10-01	NO	1.2	0.7	0.5	0.7	0.5	0.3	0.7	...	1.3	1	0.9	0.8	0.5	0.9	0.9	0.3	0.7	0.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6584	新竹	2020-12-31	THC	2.01	2.02	2	2	1.99	2	1.98	...	2.03	2.07	2.07	2.1	2.1	2.07	2.07	2.05	2.04	2.07
6585	新竹	2020-12-31	WD_HR	54	55	54	53	58	52	52	...	54	50	52	45	47	42	42	47	45	44
6586	新竹	2020-12-31	WIND_DIREC	53	52	57	58	49	54	36	...	48	43	44	33	50	40	46	46	51	38
6587	新竹	2020-12-31	WIND_SPEED	4.7	4.6	4.7	4.9	4.1	5.3	5.5	...	4.5	4.4	4.2	3.8	3.7	4.7	4.5	4.4	3.9	3.9
6588	新竹	2020-12-31	WS_HR	3.7	3.6	3.6	3.5	3.5	3.3	3.8	...	3.7	3.1	3.3	3.1	2.9	3.3	3.1	2.9	2.8	2.6

1656 rows × 27 columns

01 資料前處理 – 切割訓練集(10、11月)與測試集(12月)

訓練集

測站	日期	測項	00	01	02	03	04	05	06	...	14	15	16	17	18	19	20	21	22	23
新竹	2020-10-01	AMB_TEMP	23.7	23.8	23.8	23.9	23.9	23.8	24.1	...	29.9	29.6	28.7	27.5	26.4	25.7	25.5	25.3	24.9	24.5
新竹	2020-10-01	CH4	1.97	1.95	1.96	1.96	1.95	1.96	1.97	...	1.97	1.98	1.97	2	2.03	2.04	2.05	2.02	2.1	2.14
新竹	2020-10-01	CO	0.23	0.22	0.21	0.2	0.2	0.22	0.24	...	0.29	0.3	0.33	0.38	0.46	0.5	0.45	0.39	0.46	0.45
新竹	2020-10-01	NMHC	0.06	0.05	0.03	0.03	0.03	0.04	0.04	...	0.06	0.07	0.09	0.11	0.13	0.15	0.1	0.07	0.12	0.18
新竹	2020-10-01	NO	1.2	0.7	0.5	0.7	0.5	0.3	0.7	...	1.3	1	0.9	0.8	0.5	0.9	0.9	0.3	0.7	0.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
新竹	2020-11-30	THC	1.95	1.94	1.95	1.96	1.95	1.95	1.95	...	1.98	2	1.99	2.03	2.01	2.04	2.02	2.02	2.02	2.01
新竹	2020-11-30	WD_HR	52	53	49	50	58	55	56	...	53	52	55	51	60	45	36	47	46	39
新竹	2020-11-30	WIND_DIREC	53	50	49	47	63	54	65	...	59	43	60	56	57	41	30	55	38	41
新竹	2020-11-30	WIND_SPEED	4.4	4.2	5.2	4.8	6.1	5.4	5.8	...	5.5	5.4	4.7	5.1	5.6	5.5	5.8	5.2	4.6	4.8
新竹	2020-11-30	WS_HR	3.5	3.6	3.6	3.8	3.9	4.1	4	...	4.6	4.2	3.8	3.8	4.5	4.1	5.3	3.8	3.4	3.9

測試集

	測站	日期	測項	00	01	02	03	04	05	06	...	14	15	16	17	18	19	20	21	22	23
6031	新竹	2020-12-01	AMB_TEMP	18.5	18.4	18.3	18.2	18.2	18.3	18.4	...	21.4	20.7	20.1	19.8	19.7	19.9	20.5	20.6	20.5	20.3
6032	新竹	2020-12-01	CH4	1.95	1.95	1.95	1.95	1.95	1.95	1.94	...	1.92	1.89	1.94	1.95	1.96	1.94	1.91	1.91	1.91	1.92
6033	新竹	2020-12-01	CO	0.17	0.16	0.16	0.16	0.16	0.17	0.17	...	0.22	0.2	0.25	0.33	0.37	0.32	0.25	0.22	0.21	0.2
6034	新竹	2020-12-01	NMHC	0.06	0.05	0.06	0.06	0.04	0.06	0.03	...	0.07	0.08	0.11	0.15	0.16	0.12	0.06	0.04	0.04	0.05
6035	新竹	2020-12-01	NO	1.3	1.5	1.2	1.3	1.2	1.3	1.3	...	#	#	2.3	1.9	1.7	1.6	1.4	1.2	1	1.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6584	新竹	2020-12-31	THC	2.01	2.02	2	2	1.99	2	1.98	...	2.03	2.07	2.07	2.1	2.1	2.07	2.07	2.05	2.04	2.07
6585	新竹	2020-12-31	WD_HR	54	55	54	53	58	52	52	...	54	50	52	45	47	42	42	47	45	44
6586	新竹	2020-12-31	WIND_DIREC	53	52	57	58	49	54	36	...	48	43	44	33	50	40	46	46	51	38
6587	新竹	2020-12-31	WIND_SPEED	4.7	4.6	4.7	4.9	4.1	5.3	5.5	...	4.5	4.4	4.2	3.8	3.7	4.7	4.5	4.4	3.9	3.9
6588	新竹	2020-12-31	WS_HR	3.7	3.6	3.6	3.5	3.5	3.3	3.8	...	3.7	3.1	3.3	3.1	2.9	3.3	3.1	2.9	2.8	2.6

558 rows × 27 columns

01 資料前處理 –缺失值以及無效值以前後一小時平均值取代

對資料集做轉置成測項為欄位，以利後續做填補均值，因為接下來模型只需要數值資料，因此把多餘的欄位去除。

```
# 先對資料做轉置以利資料做後續處理
# 宣告一個新的 DataFrame 來儲存合併後的資料
train = pd.DataFrame()
test = pd.DataFrame()


for head in train_data['測項'].unique():
    train[head] = np.array(train_data[train_data['測項'] == head].iloc[:,3:]).ravel()
    test[head] = np.array(test_data[test_data['測項'] == head].iloc[:,3:]).ravel()
train
```

✓ 0.2s

	AMB_TEMP	CH4	CO	NMHC	NO	NO2	NOx	O3	PM10	PM2.5	RAINFALL	RH	SO2	THC	WD_HR	WIND_DIREC	WIND_SPEED	WS_HR
0	23.7	1.97	0.23	0.06	1.2	8	9.2	48	21	16	0	72	2	2.03	49	57	3.7	2.5
1	23.8	1.95	0.22	0.05	0.7	6	6.7	50.6	24	9	0	71	2.2	2	49	43	2.9	2.2
2	23.8	1.96	0.21	0.03	0.5	5.5	6.1	53.1	28	11	0	72	2.3	1.99	52	49	3.3	2.5
3	23.9	1.96	0.2	0.03	0.7	5.2	5.8	53	26	10	0	72	2.6	1.99	55	60	3	2.5
4	23.9	1.95	0.2	0.03	0.5	5.3	5.8	50.5	28	9	0	72	2.8	1.98	54	58	3.2	2.4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1459	19.9	1.95	0.31	0.09	1.6	8.6	10.3	33.5	16	6	0.6	74	2.5	2.04	45	41	5.5	4.1
1460	19.4	1.95	0.25	0.07	1.8	6.9	8.7	35.2	11	7	0.4	78	2	2.02	36	30	5.8	5.3
1461	18.9	1.95	0.22	0.07	1.7	6	7.8	34.9	18	9	0.6	82	2.4	2.02	47	55	5.2	3.8
1462	18.9	1.95	0.2	0.07	1.6	4.8	6.3	36.3	14	9	0.8	82	2.1	2.02	46	38	4.6	3.4
1463	18.7	1.95	0.18	0.06	1.6	4.1	5.7	37.8	18	5	0.6	82	2.1	2.01	39	41	4.8	3.9

訓練集資料

01 資料前處理 –缺失值以及無效值以前後一小時平均值取代

Step 1 – 先將無效字元，統一變為缺失值

```python
invalid_symbol = ('#', '*', 'x', 'A')


def InvalidConvertor(df):
    for row, record in df.iterrows():
        for column, record in enumerate(record):
            if record.endswith(invalid_symbol):
                df.iloc[row, column] = np.nan
    return df


train = InvalidConvertor(train)
test = InvalidConvertor(test)
```

```
1  train.isnull().sum()
✓ 0.6s
```

AMB_TEMP	0
CH4	7
CO	6
NMHC	7
NO	20
NO2	20
NOx	20
O3	9
PM10	9
PM2.5	9
RAINFALL	77
RH	0
SO2	690
THC	7
WD_HR	0
WIND_DIREC	1
WIND_SPEED	1
WS_HR	0

訓練集資料目前缺失值

01 資料前處理 –缺失值以及無效值以前後一小時平均值取代

Step 2 – 建立function把原無效字元位置利用前後平均值取代。利用到ffill、bfill方法來取前一值和後一值來算平均。

```
def FillByAvg(df):
    f = pd.DataFrame()
    b = pd.DataFrame()
    for col in df.columns:
        f[col] = df[col].fillna(method='ffill').astype('float')
        b[col] = df[col].fillna(method='bfill').astype('float')
    avg = (f + b) / 2
    return avg


train = FillByAvg(train)
test = FillByAvg(test)
```

01 資料前處理 –缺失值以及無效值以前後一小時平均值取代

Step 3 – 經由觀察還有15個值空白在最後一天的SO2，因為抓取不到後面小時的值(即都為空白值)，因此實驗過取當日其他值取中位數、平均與直接填入前一個的值6.9，得出結果皆差不多，那這裡就直接選擇放入6.9，填補這15個的空白。

```
# # 經由觀察剩下SO2最後一天的最後有15個空白，無法做平均，因此直接填入6.9
test = test.fillna('6.9')
```

```
1  test.isnull().sum()
✓ 0.4s
AMB_TEMP       0
CH4            0
CO             0
NMHC           0
NO             0
NO2            0
NOx            0
O3             0
PM10           0
PM2.5          0
RAINFALL       0
RH             0
SO2            0
THC            0
WD_HR          0
WIND_DIREC     0
WIND_SPEED     0
WS_HR          0
dtype: int64
```

```
1  train.isnull().sum()
✓ 0.5s
AMB_TEMP       0
CH4            0
CO             0
NMHC           0
NO             0
NO2            0
NOx            0
O3             0
PM10           0
PM2.5          0
RAINFALL       0
RH             0
SO2            0
THC            0
WD_HR          0
WIND_DIREC     0
WIND_SPEED     0
WS_HR          0
dtype: int64
```

缺失值處理完畢

01 資料前處理 – NR表示無降雨，以0取代

經由觀察，此次資料級，RAINFALL未有NR值，因此不須做處理。

```
1  print(train['RAINFALL'].value_counts())  # 經由觀察未有NR值
2  test['RAINFALL'].value_counts() #經由觀察未有NR值
```

✓ 0.1s

```
0.0    1355
0.1      72
0.2      16
0.4      11
0.8       5
0.6       4
2.4       1
Name: RAINFALL, dtype: int64
```

訓練集

```
0.0    664
0.2     33
0.8     11
0.4     11
0.6      9
1.0      6
1.2      3
2.2      2
1.4      2
6.2      1
2.0      1
2.6      1
Name: RAINFALL, dtype: int64
```

測試集

01 資料前處理 – 將資料轉換為row代表18種屬性,欄代表數據資料

	0	1	2	3	4	5	6	7	8	9	...	1454	1455	1456	1457	1458	1459	1460	1461	1462	1463
AMB_TEMP	23.70	23.80	23.80	23.90	23.90	23.80	24.10	24.70	26.00	27.20	...	21.60	21.50	20.40	20.00	20.10	19.90	19.40	18.90	18.90	18.70
CH4	1.97	1.95	1.96	1.96	1.95	1.96	1.97	1.97	1.96	1.98	...	1.93	1.94	1.93	1.94	1.94	1.95	1.95	1.95	1.95	1.95
CO	0.23	0.22	0.21	0.20	0.20	0.22	0.24	0.29	0.27	0.33	...	0.26	0.27	0.27	0.29	0.29	0.31	0.25	0.22	0.20	0.18
NMHC	0.06	0.05	0.03	0.03	0.03	0.04	0.04	0.05	0.06	0.07	...	0.05	0.06	0.06	0.09	0.07	0.09	0.07	0.07	0.07	0.06
NO	1.20	0.70	0.50	0.70	0.50	0.30	0.70	0.90	1.00	1.80	...	2.50	2.40	2.00	1.80	1.60	1.60	1.80	1.70	1.60	1.60
NO2	8.00	6.00	5.50	5.20	5.30	5.80	8.00	7.60	6.60	8.00	...	4.50	5.40	6.60	9.00	7.50	8.60	6.90	6.00	4.80	4.10
NOx	9.20	6.70	6.10	5.80	5.80	6.30	8.60	8.50	7.60	9.80	...	6.90	7.70	8.50	10.80	9.10	10.30	8.70	7.80	6.30	5.70
O3	48.00	50.60	53.10	53.00	50.50	47.80	44.80	46.60	51.90	55.80	...	42.40	39.70	35.90	32.40	34.50	33.50	35.20	34.90	36.30	37.80
PM10	21.00	24.00	28.00	26.00	28.00	22.00	26.00	27.00	29.00	23.00	...	23.00	30.00	15.00	14.00	14.00	16.00	11.00	18.00	14.00	18.00
PM2.5	16.00	9.00	11.00	10.00	9.00	15.00	10.00	10.00	10.00	9.00	...	6.00	9.00	5.00	3.00	4.00	6.00	7.00	9.00	9.00	5.00
RAINFALL	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.20	0.20	0.40	0.40	0.40	0.60	0.40	0.60	0.80	0.60
RH	72.00	71.00	72.00	72.00	72.00	72.00	72.00	68.00	61.00	55.00	...	60.00	60.00	69.00	72.00	72.00	74.00	78.00	82.00	82.00	82.00
SO2	2.00	2.20	2.30	2.60	2.80	3.00	3.40	2.90	2.50	2.40	...	2.20	2.80	3.40	3.30	2.80	2.50	2.00	2.40	2.10	2.10
THC	2.03	2.00	1.99	1.99	1.98	2.00	2.01	2.02	2.02	2.05	...	1.98	2.00	1.99	2.03	2.01	2.04	2.02	2.02	2.02	2.01
WD_HR	49.00	49.00	52.00	55.00	54.00	54.00	55.00	46.00	48.00	47.00	...	53.00	52.00	55.00	51.00	60.00	45.00	36.00	47.00	46.00	39.00
WIND_DIREC	57.00	43.00	49.00	60.00	58.00	47.00	54.00	46.00	57.00	38.00	...	59.00	43.00	60.00	56.00	57.00	41.00	30.00	55.00	38.00	41.00
WIND_SPEED	3.70	2.90	3.30	3.00	3.20	2.50	2.90	3.10	4.20	4.30	...	5.50	5.40	4.70	5.10	5.60	5.50	5.80	5.20	4.60	4.80
WS_HR	2.50	2.20	2.50	2.50	2.40	2.30	2.10	2.40	3.10	3.40	...	4.60	4.20	3.80	3.80	4.50	4.10	5.30	3.80	3.40	3.90

18 rows × 1464 columns

轉置後的訓練集資料 (18 rows 1464 columns)

02 時間序列 – PM2.5與所有屬性資料準備

將未來第一個小時為預測目標，所以要 i + 5 + 1(i+6)

```
pm25_1hr_train_X = np.array([list(train_data.loc['PM2.5', i:i+5])
                             for i in range(len(train_data.columns)-6)])
pm25_1hr_train_y = np.array([train_data.loc['PM2.5', i+6]  # 5 + 1 預測未來一個小時
                             for i in range(len(train_data.columns)-6)])
```

將未來第六個小時為預測目標，所以要 i + 5 + 6(i+11)

```
pm25_6hr_train_X = np.array([list(train_data.loc['PM2.5', i:i+5])
                             for i in range(len(train_data.columns)-11)])
pm25_6hr_train_y = np.array([train_data.loc['PM2.5', i+11]  # 5 + 6 預測未來6個小時
                             for i in range(len(train_data.columns)-11)])
```

測試集資料與所有屬性資料準備皆與上述大同小異，因此不再放截圖，以免畫面太冗餘。

03 模型訓練與結果 – LinearRegression 調整參數與結果

```python
# Linear Regression - PM2.5
LR_pm25_1hr = LinearRegression(fit_intercept=False, normalize=True)
LR_pm25_1hr.fit(pm25_1hr_train_X, pm25_1hr_train_y)
LR_pm25_1hr_pred = LR_pm25_1hr.predict(pm25_1hr_test_X)
print("MAE of Linear Regression - pm2.5 - 1hr PM2.5: ",
      mean_absolute_error(pm25_1hr_test_y, LR_pm25_1hr_pred))

LR_pm25_6hr = LinearRegression(
    fit_intercept=False, normalize=True)
LR_pm25_6hr.fit(pm25_6hr_train_X, pm25_6hr_train_y)
LR_pm25_6hr_pred = LR_pm25_6hr.predict(pm25_6hr_test_X)
print("MAE of Linear Regression - pm2.5 - 6hr PM2.5: ",
      mean_absolute_error(pm25_6hr_test_y, LR_pm25_6hr_pred))


# Linear Regression - 18 Attributes
LR_all_1hr = LinearRegression(
    fit_intercept=False, normalize=True)
LR_all_1hr.fit(all_1hr_train_X, all_1hr_train_y)
LR_all_1hr_pred = LR_all_1hr.predict(all_1hr_test_X)
print("MAE of Linear Regression - all - 1hr PM2.5: ",
      mean_absolute_error(all_1hr_test_y, LR_all_1hr_pred))

LR_all_6hr = LinearRegression(
    fit_intercept=False, normalize=True)
LR_all_6hr.fit(all_6hr_train_X, all_6hr_train_y)
LR_all_6hr_pred = LR_all_6hr.predict(all_6hr_test_X)
print("MAE of Linear Regression - all - 6hr PM2.5: ",
      mean_absolute_error(all_6hr_test_y, LR_all_6hr_pred))
```

```
MAE of Linear Regression - pm2.5 - 1hr PM2.5:  2.4263326461741257
MAE of Linear Regression - pm2.5 - 6hr PM2.5:  4.349127026907553
MAE of Linear Regression - all - 1hr PM2.5:  2.6958862086920634
MAE of Linear Regression - all - 6hr PM2.5:  5.635087016008428
```

03 模型訓練與結果 – XGBoost 調整參數

XGBoost 參數大多是數值型，因此這裡使用到GridSearchCV來幫忙調整參數

```
cv_params = {'min_child_weight': np.linspace(1, 10, 10, dtype=int)}
XGB_all_1hr = XGBRegressor(**other_params)
gs = GridSearchCV(XGB_all_1hr, cv_params, verbose=2,
                  refit=True, cv=5, n_jobs=-1)
gs.fit(all_1hr_train_X, all_1hr_train_y)
print("參數的最佳取值: ", gs.best_params_)
print("最佳模型得分: ", gs.best_score_)
```

5.9s

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
參數的最佳取值:  {'min_child_weight': 5}
最佳模型得分:  0.6689032394376048
```

```
other_params = {'eta': 0.3, 'n_estimators': 100, 'gamma': 0, 'max_depth': 1, 'min_child_weight': 5,
                'colsample_bytree': 1, 'colsample_bylevel': 1, 'subsample': 1, 'reg_lambda': 1, 'reg_alpha'
                'seed': 33}

cv_params = {'gamma': np.linspace(0, 0.1, 12, dtype=int)}
XGB_all_1hr = XGBRegressor(**other_params)
gs = GridSearchCV(XGB_all_1hr, cv_params, verbose=2,
                  refit=True, cv=5, n_jobs=-1)
gs.fit(all_1hr_train_X, all_1hr_train_y)
print("參數的最佳取值: ", gs.best_params_)
print("最佳模型得分: ", gs.best_score_)
```

7.3s

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
參數的最佳取值:  {'gamma': 0}
最佳模型得分:  0.6689032394376048
```

參數調整畫面

```
params_pm25_1hr = {'eta': 0.3, 'n_estimators': 100, 'gamma': 0, 'max_depth': 1, 'min_child_weight': 8,
                   'colsample_bytree': 1, 'colsample_bylevel': 1, 'subsample': 1, 'reg_lambda': 60, 'reg_alpha': 6,
                   'seed': 33}


params_pm25_6hr = {'eta': 0.0774263682681127, 'n_estimators': 100, 'gamma': 0, 'max_depth': 1, 'min_child_weight': 3,
                   'colsample_bytree': 1, 'colsample_bylevel': 1, 'subsample': 1, 'reg_lambda': 60, 'reg_alpha': 4,
                   'seed': 33}

params_all_1hr = {'eta': 0.1291549665014884, 'n_estimators': 200, 'gamma': 0, 'max_depth': 1, 'min_child_weight': 1,
                  'colsample_bytree': 1, 'colsample_bylevel': 1, 'subsample': 1, 'reg_lambda': 60, 'reg_alpha': 4,
                  'seed': 33}

params_all_6hr = {'eta': 0.046415888336127774, 'n_estimators': 100, 'gamma': 0, 'max_depth': 5, 'min_child_weight': 1,
                  'colsample_bytree': 1, 'colsample_bylevel': 1, 'subsample': 1, 'reg_lambda': 10, 'reg_alpha': 6,
                  'seed': 33}
```

0.4s

XGBoost不同預測參數調整結果

03 模型訓練與結果 – XGBoost結果

```python
XGB_pm25_6hr_pred = XGB_pm25_6hr.predict(pm25_6hr_test_X)
print("MAE of XGBoost - pm2.5 - 6hr PM2.5: ",
      mean_absolute_error(pm25_6hr_test_y, XGB_pm25_6hr_pred))


# XGBoost - 18 Attributes

XGB_all_1hr = XGBRegressor(**params_all_1hr)
XGB_all_1hr.fit(all_1hr_train_X, all_1hr_train_y)
XGB_all_1hr_pred = XGB_all_1hr.predict(all_1hr_test_X)
print("MAE of XGBoost - all - 1hr PM2.5:",
      mean_absolute_error(all_1hr_test_y, XGB_all_1hr_pred))

XGB_all_6hr = XGBRegressor(**params_all_6hr)
XGB_all_6hr.fit(all_6hr_train_X, all_6hr_train_y)
XGB_all_6hr_pred = XGB_all_6hr.predict(all_6hr_test_X)
print("MAE of XGBoost - all - 6hr PM2.5: ",
      mean_absolute_error(all_6hr_test_y, XGB_all_6hr_pred))

```

✓ 1.6s

```
MAE of XGBoost - pm2.5 - 1hr PM2.5:  2.654466922690229
MAE of XGBoost - pm2.5 - 6hr PM2.5:  4.770751389117065
MAE of XGBoost - all - 1hr PM2.5: 2.618156887974519
MAE of XGBoost - all - 6hr PM2.5:  4.436619846427262
```

03 模型訓練與結果 – MAE 表格整理

Linear Regression	預測未來一小時	預測未來六小時
PM2.5	2.43	4.35
所有屬性	2.7	5.64

XGBoost	預測未來一小時	預測未來六小時
PM2.5	2.65	4.77
所有屬性	2.62	4.44