

結果：

	將未來第一個小時 當預測目標，X 只有 PM2.5	將未來第六個小時 當預測目標 X 只有 PM2.5	將未來第一個小時 當預測目標， x 取 所有 18 種屬性	將未來第六個小時 當預測目標 X 有所 有 18 種屬性
Linear Regression	2.6136558911044343	4.839609070011748	3.7329306223569962	3.7604398413161895
Random Forest Regression	3.285230352303523	5.560027285129604	3.5657181571815717	3.690995907230559

1. 載入資料，因為資料是中文編碼，utf-8 會有亂碼，所以用 big5

```
In [1]: import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.ensemble import RandomForestClassifier
data = pd.read_csv('新竹_2019.csv', encoding='big5')
data
```

4	新竹	2019/1/1 0:00	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
...
6476	新竹	2019/12/31 0:00	THC	1.81	1.79	1.82	1.84	1.84	1.85	1.85	...	1.89	1.9	1.89	1.9	1.88	1.84	1.81	1.81	1.81	1.8
6477	新竹	2019/12/31 0:00	WD_HR	42	41	40	41	38	42	41	...	34	40	40	49	48	51	57	50	49	53
6478	新竹	2019/12/31 0:00	WIND_DIREC	35	42	36	47	41	35	35	...	26	44	46	35	50	47	64	40	55	52
6479	新竹	2019/12/31 0:00	WIND_SPEED	3.7	3.5	3.4	3.7	4.6	5	4.8	...	3.6	5	3.9	4.7	5.1	4.4	4.1	4.4	3.7	4.6
6480	新竹	2019/12/31 0:00	WS_HR	2.6	2.7	2.9	3.6	3.4	3.5	3.5	...	3.5	3.6	3.4	3.6	3.5	3.7	3.5	3.1	3.2	3.6

6481 rows x 27 columns

2. 清理資料

```
In [2]: data.drop(index = [0], axis=0, inplace=True) # row : axis = 0, column : axis = 1
data
```

```
Out[2]:
```

	測站	日期	測項	0	1	2	3	4	5	6	...	14	15	16	17	18	19	20	21	22	23
1	新竹	2019/1/1 0:00	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	17.5	17.7	17.5	
2	新竹	2019/1/1 0:00	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	2019/1/1 0:00	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	2019/1/1 0:00	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	新竹	2019/1/1 0:00	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
...
6476	新竹	2019/12/31 0:00	THC	1.81	1.79	1.82	1.84	1.84	1.85	1.85	...	1.89	1.9	1.89	1.9	1.88	1.84	1.81	1.81	1.81	1.8
6477	新竹	2019/12/31 0:00	WD_HR	42	41	40	41	38	42	41	...	34	40	40	49	48	51	57	50	49	53
6478	新竹	2019/12/31 0:00	WIND_DIREC	35	42	36	47	41	35	35	...	26	44	46	35	50	47	64	40	55	52
6479	新竹	2019/12/31 0:00	WIND_SPEED	3.7	3.5	3.4	3.7	4.6	5	4.8	...	3.6	5	3.9	4.7	5.1	4.4	4.1	4.4	3.7	4.6

(1) 處理前後空白字元

```
In [4]: data.columns
```

```
Out[4]: Index(['測站', '日期', '測項', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23'],
              dtype='object')
```

```
In [5]: data.columns = data.columns.str.strip()
data.columns
```

```
Out[5]: Index(['測站', '日期', '測項', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23'],
              dtype='object')
```

(2) 拿 10 月之後的資料

```
In [12]: data = data.loc[data.loc[data['日期'] == '2019/10/1 0:00'].index[0] :, :]
```

```
In [13]: data.head(30)
```

Out [13]:

	測站	日期	測項	0	1	2	3	4	5	6	...	14	15	16	17	18	19	20	21	22	23
4825	新竹	2019/10/1 0:00	AMB_TEMP	24.7	25.1	25.4	25.5	25.3	25.1	24.6	...	32.4	31.9	30.4	29.1	28.7	28.3	28	27.4	27	26.5
4826	新竹	2019/10/1 0:00	CH4	1.66	1.66	1.7	1.71	1.72	1.71	1.75	...	1.69	1.72	1.74	1.74	1.78	1.82	1.82	1.83	1.93	1.96
4827	新竹	2019/10/1 0:00	CO	0.05	0.13	0.15	0.17	0.16	0.16	0.22	...	0.27	0.32	0.36	0.39	0.49	0.57	0.58	0.6	0.69	0.49
4828	新竹	2019/10/1 0:00	NMHC	0	0	0	0.02	0.02	0.01	0.03	...	0.07	0.1	0.08	0.08	0.13	0.21	0.22	0.21	0.22	0.17
4829	新竹	2019/10/1 0:00	NO	0	0.3	0.3	0.3	0.3	0.3	1.2	...	1.6	1.1	0.8	0.6	0.6	0.7	0.6	2	5.1	3.8
4830	新竹	2019/10/1 0:00	NO2	1.3	1.1	1.3	1.8	2.1	2.3	6.1	...	5.9	8.3	10.1	9.6	14.9	19.5	22.2	26.7	24	15.7
4831	新竹	2019/10/1 0:00	NOx	1.2	1.2	1.7	2.1	2.4	2.7	7.3	...	7.4	9.4	10.8	10.3	15.5	20.2	22.8	28.6	29.1	19.5
4832	新竹	2019/10/1 0:00	O3	16.7	17.3	21.9	21.5	18.8	17.5	11.8	...	41	54.1	57.7	49.3	39.1	30.8	23.3	13.1	7.5	5.5
4833	新竹	2019/10/1 0:00	PM10	18	18	29	29	27	19	23	...	17	23	47	53	47	35	33	21	22	14
4834	新竹	2019/10/1 0:00	PM2.5	2	4	6	9	10	7	4	...	3	8	19	17	19	19	19	14	17	8

(3) NR 表示無降雨，以 0 取代

```
In [14]: data = data.replace(['NR'], [0.0])
data.head(30)
```

(4) 把怪怪的字元拿掉，用前後兩個值的平均

```
In [18]: symbols = ['#', '*', 'x', 'A']
for i in range(len(data)): # row
    for j in range(3, len(data.columns)): # column
        for symbol in symbols:
            if symbol in data.iloc[i, j]:
                front_j = j - 1
                back_j = j + 1
                front_i = i
                back_i = i
                if front_j == 2: # 代表取到了測項，要往前一天的最後一小時取資料
                    front_j = len(data.columns) - 1
                    front_i = i - 18 # 前一天的 row index在前面18個
                if back_j == len(data.columns): # 代表取到超過今天的數據了
                    back_j = 3 # 隔天的第一筆數據
                    back_i = i + 18
                # 下一個的symbol不一定是上一個index的symbol，所以要再重測。
                # 因為我是用row by row, column by column，所以不需要去測前面的
                while True:
                    s = 0 # 用來記錄是否都不是symbol
                    for symbol in symbols:
                        if symbol in data.iloc[back_i, back_j]:
                            back_j += 1
                        else:
                            s += 1
                    if s == 4: # 代表4個symbol他都不在back裡面了
                        break
                # 因為可以讓中間都是異常值的數直接被平均數替掉
                data.iloc[i, j:back_j] = str((float(data.iloc[front_i, front_j]) + float(data.iloc[back_i, back_j])))
```

怪字元拿掉了！！

掃描 data 的方式由左至右、由上至下

這邊是分三種狀況：

- 如果取到了‘測項’那欄，那就要回前 18 個列取前一天 24 時的資料。
- 如果取到了超過 data 欄位的資料，那就要去之後 18 個列取下一天 0 時的資料。
- 剩下的狀況就是目前有異常值，但還要繼續檢查下一個是不是異常值。

(5) 將資料切割成訓練集(10.11 月)以及測試集(12 月)

```
In [20]: data.loc[data['日期'] == '2019/12/31 0:00'].index[0]
Out[20]: 6463

In [21]: training_set = data.loc[data.loc[data['日期'] == '2019/10/1 0:00'].index[0] :
      data.loc[data['日期'] == '2019/12/1 0:00'].index[0] - 1, :]
training_set

Out[21]:
```

	測站	日期	測項	0	1	2	3	4	5	6	...	14	15	16	17	18	19	20	21	22	23
4825	新竹	2019/10/1 0:00	AMB_TEMP	24.7	25.1	25.4	25.5	25.3	25.1	24.6	...	32.4	31.9	30.4	29.1	28.7	28.3	28	27.4	27	26.5
4826	新竹	2019/10/1 0:00	CH4	1.66	1.66	1.7	1.71	1.72	1.71	1.75	...	1.69	1.72	1.74	1.74	1.78	1.82	1.82	1.83	1.93	1.96
4827	新竹	2019/10/1 0:00	CO	0.05	0.13	0.15	0.17	0.16	0.16	0.22	...	0.27	0.32	0.36	0.39	0.49	0.57	0.58	0.6	0.69	0.49
4828	新竹	2019/10/1 0:00	NMHC	0	0	0	0.02	0.02	0.01	0.03	...	0.07	0.1	0.08	0.08	0.13	0.21	0.22	0.21	0.22	0.17
4829	新竹	2019/10/1 0:00	NO	0	0.3	0.3	0.3	0.3	0.3	1.2	...	1.6	1.1	0.8	0.6	0.6	0.7	0.6	2	5.1	3.8
...
5918	新竹	2019/11/30 0:00	THC	1.84	1.82	1.84	1.84	1.91	1.97	1.9	...	1.83	1.94	1.95	1.94	2.04	2.23	2.2	2	2.05	1.94
5919	新竹	2019/11/30 0:00	WD_HR	56	65	67	66	95	96	41	...	281	271	257	246	213	165	198	162	167	168
5920	新竹	2019/11/30 0:00	WIND_DIREC	53	70	98	20	94	83	39	...	258	273	244	167	178	162	225	170	115	179
5921	新竹	2019/11/30 0:00	WIND_SPEED	1.7	1.7	0.8	0.7	0.9	1.3	1.3	...	2.1	1.7	1.3	0.8	0.9	1.2	0.8	0.9	0.9	1
5922	新竹	2019/11/30 0:00	WS_HR	1.8	1.3	0.9	0.4	0.7	0.8	0.9	...	1.5	1.6	1.5	1.3	0.6	0.9	0.5	0.6	0.6	1

1098 rows x 27 columns

(6) 再把每小時的資料從 object transfer to float，testing_set 也做一樣的事

```
In [24]: for column in training_set.columns[3:]:
      training_set[column] = training_set[column].astype(float)
training_set.dtypes

Out[24]: 測站      object
日期      object
測項      object
0         float64
1         float64
2         float64
3         float64
4         float64
5         float64
6         float64
7         float64
8         float64
9         float64
10        float64
11        float64
12        float64
13        float64
14        float64
15        float64
16        float64

In [25]: testing_set = data.loc[data.loc[data['日期'] == '2019/12/1 0:00'].index[0]: , :]
testing_set
```

5924	新竹	2019/12/1 0:00	CH4	1.86	1.91	1.89	1.87	1.96	1.86	1.86	...	1.73	1.74	1.75	1.78	1.77	1.75	1.77	1.81	1.79	1.79
5925	新竹	2019/12/1 0:00	CO	0.28	0.29	0.24	0.2	0.23	0.22	0.3	...	0.23	0.26	0.28	0.36	0.37	0.32	0.29	0.32	0.27	0.29
5926	新竹	2019/12/1 0:00	NMHC	0.08	0.1	0.09	0.09	0.1	0.08	0.11	...	0.04	0.05	0.07	0.12	0.12	0.08	0.09	0.12	0.09	0.09
5927	新竹	2019/12/1 0:00	NO	0.6	0.3	0.3	0.6	0.3	1.95	1.95	...	0.3	0.5	0.3	0.3	0.3	0.3	0.3	0.3	0.4	0.5
...
6476	新竹	2019/12/31 0:00	THC	1.81	1.79	1.82	1.84	1.84	1.85	1.85	...	1.89	1.9	1.89	1.9	1.88	1.84	1.81	1.81	1.81	1.8
6477	新竹	2019/12/31 0:00	WD_HR	42	41	40	41	38	42	41	...	34	40	40	49	48	51	57	50	49	53
6478	新竹	2019/12/31 0:00	WIND_DIREC	35	42	36	47	41	35	35	...	26	44	46	35	50	47	64	40	55	52
6479	新竹	2019/12/31 0:00	WIND_SPEED	3.7	3.5	3.4	3.7	4.6	5	4.8	...	3.6	5	3.9	4.7	5.1	4.4	4.1	4.4	3.7	4.6
6480	新竹	2019/12/31 0:00	WS_HR	2.6	2.7	2.9	3.6	3.4	3.5	3.5	...	3.5	3.6	3.4	3.6	3.5	3.7	3.5	3.1	3.2	3.6

558 rows x 27 columns

```
In [27]: for column in testing_set.columns[3:]:
testing_set[column] = testing_set[column].astype(float)
testing_set.dtypes

Out[27]: 測站      object
日期      object
測項      object
0         float64
1         float64
2         float64
3         float64
4         float64
5         float64
6         float64
7         float64
8         float64
9         float64
10        float64
11        float64
12        float64
13        float64
14        float64
15        float64
16        float64
```

(7) 製作時序資料: 將資料形式轉換為行(row)代表 18 種屬性，欄(column)代表逐時數據資料 hint: 將訓練集每 18 行合併，轉換成維度為(18,6124)的 DataFrame(每個屬性都有 61 天 24 小時共 1464 筆資料)

Training_set

```
In [29]: # 有61筆data，i處理到 len(data) - 2 = 59就好
for i in range(int(len(training_set) / 18) - 1):
    new_training_set = pd.merge(new_training_set, training_set.iloc[(i + 1) * 18 : (i + 2) * 18, 2 : ],on='測項')
new_training_set
```

12	新竹	2019/10/1 0:00	SO2	1.10	1.30	1.40	1.50	1.50	1.60	1.80	...	1.90	4.60	3.90	2.30	3.30	3.50	3.00	3.10	2.40	2.20
13	新竹	2019/10/1 0:00	THC	1.65	1.65	1.70	1.73	1.74	1.72	1.78	...	1.83	1.94	1.95	1.94	2.04	2.23	2.20	2.00	2.05	1.94
14	新竹	2019/10/1 0:00	WD_HR	292.00	286.00	291.00	286.00	268.00	277.00	247.00	...	281.00	271.00	257.00	246.00	213.00	165.00	198.00	162.00	167.00	168.00
15	新竹	2019/10/1 0:00	WIND_DIREC	283.00	289.00	276.00	289.00	250.00	262.00	239.00	...	258.00	273.00	244.00	167.00	178.00	162.00	225.00	170.00	115.00	179.00
16	新竹	2019/10/1 0:00	WIND_SPEED	2.10	1.40	2.00	2.50	1.80	2.40	2.00	...	2.10	1.70	1.30	0.80	0.90	1.20	0.80	0.90	0.90	1.00
17	新竹	2019/10/1 0:00	WS_HR	0.90	1.10	1.80	2.20	1.80	1.90	2.00	...	1.50	1.60	1.50	1.30	0.60	0.90	0.50	0.60	0.60	1.00

18 rows x 1467 columns

再把不想留的 column 拿掉

```
In [30]: new_training_set.drop(['測站', '日期'], axis=1, inplace=True)
new_training_set
```

7	O3	16.70	17.30	21.90	21.50	18.80	17.50	11.80	8.20	10.40	...	64.10	52.10	48.10	51.80	37.00	21.00	15.40	23.00	16.40	20.80
8	PM10	18.00	18.00	29.00	29.00	27.00	19.00	23.00	13.00	10.00	...	30.00	51.00	41.00	38.00	42.00	56.00	56.00	51.00	41.00	47.00
9	PM2.5	2.00	4.00	6.00	9.00	10.00	7.00	4.00	6.00	7.00	...	16.00	35.00	17.00	20.00	22.00	34.00	36.00	32.00	27.00	22.00
10	RAINFALL	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11	RH	89.00	87.00	85.00	84.00	85.00	86.00	88.00	90.00	85.00	...	54.00	54.00	65.00	75.00	78.00	79.00	81.00	83.00	84.00	85.00
12	SO2	1.10	1.30	1.40	1.50	1.50	1.60	1.80	2.00	1.90	...	1.90	4.60	3.90	2.30	3.30	3.50	3.00	3.10	2.40	2.20
13	THC	1.65	1.65	1.70	1.73	1.74	1.72	1.78	1.84	1.81	...	1.83	1.94	1.95	1.94	2.04	2.23	2.20	2.00	2.05	1.94
14	WD_HR	292.00	286.00	291.00	286.00	268.00	277.00	247.00	203.00	193.00	...	281.00	271.00	257.00	246.00	213.00	165.00	198.00	162.00	167.00	168.00
15	WIND_DIREC	283.00	289.00	276.00	289.00	250.00	262.00	239.00	183.00	187.00	...	258.00	273.00	244.00	167.00	178.00	162.00	225.00	170.00	115.00	179.00
16	WIND_SPEED	2.10	1.40	2.00	2.50	1.80	2.40	2.00	1.70	2.30	...	2.10	1.70	1.30	0.80	0.90	1.20	0.80	0.90	0.90	1.00
17	WS_HR	0.90	1.10	1.80	2.20	1.80	1.90	2.00	1.00	1.50	...	1.50	1.60	1.50	1.30	0.60	0.90	0.50	0.60	0.60	1.00

18 rows x 1465 columns

```
In [32]: for i in range(int(len(testing_set) / 18) - 1):
new_testing_set = pd.merge(new_testing_set, testing_set.iloc[(i + 1) * 18 : (i + 2) * 18, 2 : ],on='測項')
new_testing_set
```

3. 將未來第一個小時當預測目標，X 只有 PM2.5 (e.g. X[0]會有 6 個特徵，即第 0~5 小時的 PM2.5 數值)

```
In [35]: training_x = new_training_set.iloc[new_training_set.loc[new_training_set['測項'] == 'PM2.5'].index[0, 1:]]
print(training_x)
print(training_x.shape)
print(len(training_x))

0_x      2
1_x      4
2_x      6
3_x      9
4_x     10
..
19     34
20     36
21     32
22     27
23     22
Name: 9, Length: 1464, dtype: object
(1464,)
1464

In [36]: training_x_np = np.zeros((len(training_x) - 6, 6))
print(training_x_np)
print(training_x_np.shape[0])

[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 ...
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
1458
```

```
In [37]: def get_PM25_data_x_from(data_df, data_np, amount_in_group):
# data_df : 原本的資料
# data_np : 我要存的格式
# amount_in_group : 幾個資料一組
for i in range(len(data_df) - amount_in_group):
    k = i
    for j in range(6):
        data_np[i][j] = data_df.iloc[k, j]
        k += 1
    return data_np
training_x_np = get_PM25_data_x_from(training_x, training_x_np, 6)
print(training_x_np)
print(training_x_np[1450 : ])

[[ 2.  4.  6.  9. 10.  7.]
 [ 4.  6.  9. 10.  7.  4.]
 [ 6.  9. 10.  7.  4.  6.]
 ...
 [35. 17. 20. 22. 34. 36.]
 [17. 20. 22. 34. 36. 32.]
 [20. 22. 34. 36. 32. 27.]]
[[ 8. 14. 18. 15. 16. 35.]
 [14. 18. 15. 16. 35. 17.]
 [18. 15. 16. 35. 17. 20.]
 [15. 16. 35. 17. 20. 22.]
 [16. 35. 17. 20. 22. 34.]
 [35. 17. 20. 22. 34. 36.]
 [17. 20. 22. 34. 36. 32.]
 [20. 22. 34. 36. 32. 27.]]
```

```
In [38]: training_y_np = np.zeros(len(training_x_np))
def get_PM25_data_y_from(data_df, data_np, amount_in_group):
    for i in range(len(data_df) - amount_in_group):
        data_np[i] = data_df.iloc[i + amount_in_group,]
    return data_np
training_y_np = get_PM25_data_y_from(training_x, training_y_np, 6)
print(training_y_np)
print(len(training_y_np))

[ 4.  6.  7. ... 32. 27. 22.]
1458
```

(2) 這裡整理 testing_set 的 x, y 資料

```
In [39]: testing_x = new_testing_set.iloc[new_testing_set.loc[new_testing_set['測項'] == 'PM2.5'].index[0], 1:]
testing_x_np = np.zeros((len(testing_x) - 6, 6))
testing_x_np = get_PM25_data_x_from(testing_x, testing_x_np, 6)
print(testing_x_np)
print(len(testing_x_np))

[[18. 18. 18. 16. 18. 18.]
 [18. 18. 16. 18. 18. 15.]
 [18. 16. 18. 18. 15. 26.]
 ...
 [18. 20. 17. 18. 12. 13.]
 [20. 17. 18. 12. 13. 10.]
 [17. 18. 12. 13. 10. 12.]]
738
```

```
In [40]: testing_y_np = np.zeros(len(testing_x_np))
testing_y_np = get_PM25_data_y_from(testing_x, testing_y_np, 6)
print(testing_y_np)
print(len(testing_y_np))

8.  6.  6.  16. 15. 17.  8. 11. 17. 17.  8. 11. 10. 14.
12. 11. 19. 22. 14. 16. 19. 17. 18. 15. 13.  8.  9. 14.
10.  7.  3.  4.  4.  5.  6.  4.  3.  5.  4.  7.  9.  9.
11. 10.  6.  6.  8. 11.  8.  9.  8. 13. 11. 10.  8.  8.
10. 13. 12. 13. 13. 12. 16. 12. 11. 18. 15. 15. 15. 13.
13. 14. 11.  9. 17. 13. 12.  9.  7.  8. 11. 11. 13.  9.
 9. 10. 10.  9.  8.  8. 13. 15. 13. 15. 13. 15. 13. 16.
15. 20. 19. 10. 21. 22. 24. 30. 31. 32. 30. 31. 26. 28.
29. 31. 30. 11. 19. 11. 14. 18. 21. 21. 19. 15. 12. 12.
 8. 13. 14. 11.  7.  9.  9. 11.  9.  9. 19. 20. 20. 20.
21. 22. 23. 24. 23. 18. 20. 17. 19. 16. 18. 13. 11.  9.
12. 12.  9.  8.  7.  7.  8.  5.  6.  6.  6.  7.  6.  8.
12.  8. 11. 13. 12.  9.  7.  8. 12. 14. 17. 19. 25. 27.
24. 21. 21.  6. 15. 16. 15. 13. 13. 14. 14. 16. 15. 14.
15. 12. 12. 10. 13. 12. 10. 12.  7.  3.  3.  1.  1.  1.
 1.  6. 15. 12. 13.  9.  6.  5.  3.  5.  6.  8.  7.
 7.  6.  5.  7.  9. 10. 11. 10.  8.  8.  7. 10. 11. 16.
14. 18. 20. 17. 18. 12. 13. 10. 12. 11. ]
738
```

(3) 做 Linear Regression

''' The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0. '''

```
In [41]: def linear_regression(training_set_x, training_set_y, testing_set_x, testing_set_y):  
         x, y = training_set_x, training_set_y  
         reg = LinearRegression().fit(x, y)  
         predict_y = reg.predict(testing_set_x)  
         y_true = testing_set_y  
         y_pred = predict_y  
         return mean_absolute_error(y_true, y_pred)
```

''' 用於評估預測結果和真實資料集的接近程度的程度，其值越小說明擬合效果越好。MAE output is non-negative floating point. The best value is 0.0. '''

```
In [42]: linear_regression(training_x_np, training_y_np, testing_x_np, testing_y_np)
```

```
Out[42]: 2.6136558911044343
```

(4) 做 Random Forest Regression

```
In [43]: def random_forest_regression(training_set_x, training_set_y, testing_set_x, testing_set_y):  
         clf = RandomForestClassifier(max_depth=10, random_state=1)  
         x, y = training_set_x, training_set_y  
         y = np.array(y, dtype=int)  
         clf.fit(x, y)  
         predict_y = clf.predict(testing_set_x)  
         y_true = testing_set_y  
         y_pred = predict_y  
         return mean_absolute_error(y_true, y_pred)
```

```
In [44]: random_forest_regression(training_x_np, training_y_np, testing_x_np, testing_y_np)
```

```
Out[44]: 3.285230352303523
```


4. 將未來第六個小時當預測目標 X 只有 PM2.5 (e.g. X[0]會有 6 個特徵，即第 0~5 小時的 PM2.5 數值)

(1) 這裡整理 training_set 的 x, y 資料

```
In [45]: training_x_np = np.zeros((len(training_x) - 11, 6))
print(len(training_x_np))
training_x_np = get_PM25_data_x_from(training_x, training_x_np, 11)
print(training_x_np)
1453
[[ 2.  4.  6.  9. 10.  7.]
 [ 4.  6.  9. 10.  7.  4.]
 [ 6.  9. 10.  7.  4.  6.]
 ...
 [ 8. 14. 18. 15. 16. 35.]
 [14. 18. 15. 16. 35. 17.]
 [18. 15. 16. 35. 17. 20.]]

In [46]: training_y_np = np.zeros(len(training_x_np))
training_y_np = get_PM25_data_y_from(training_x, training_y_np, 11)
print(training_y_np)
[ 2.  3.  4. ... 32. 27. 22.]
```

(2) 這裡整理 testing_set 的 x, y 資料

```
In [47]: testing_x = new_testing_set.iloc[new_testing_set.loc[new_testing_set['測項'] == 'PM2.5'].index[0], 1:]
testing_x_np = np.zeros((len(testing_x) - 11, 6))
testing_x_np = get_PM25_data_x_from(testing_x, testing_x_np, 11)
print(testing_x_np)
print(len(testing_x_np))
[[18. 18. 18. 16. 18. 18.]
 [18. 18. 16. 18. 18. 15.]
 [18. 16. 18. 18. 15. 26.]
 ...
 [ 7. 10. 11. 16. 14. 18.]
 [10. 11. 16. 14. 18. 20.]
 [11. 16. 14. 18. 20. 17.]]
733

In [48]: testing_y_np = np.zeros(len(testing_x_np))
testing_y_np = get_PM25_data_y_from(testing_x, testing_y_np, 11)
print(testing_y_np)
print(len(testing_y_np))
17.  8. 11. 17. 17.  8. 11. 10. 14. 12. 11. 19. 22. 14.
16. 19. 17. 18. 15. 13.  8.  9. 14. 10.  7.  3.  4.  4.
 5.  6.  4.  3.  5.  4.  7.  9.  9. 11. 10.  6.  6.  8.
11.  8.  9.  8. 13. 11. 10.  8.  8. 10. 13. 12. 13. 13.
12. 16. 12. 11. 18. 15. 15. 15. 13. 13. 14. 11.  9. 17.
13. 12.  9.  7.  8. 11. 11. 13.  9.  9. 10. 10.  9.  8.
 8. 13. 15. 13. 15. 13. 15. 13. 16. 15. 20. 19. 10. 21.
22. 24. 30. 31. 32. 30. 31. 26. 28. 29. 31. 30. 11. 19.
11. 14. 18. 21. 21. 19. 15. 12. 12.  8. 13. 14. 11.  7.
 9.  9. 11.  9.  9. 19. 20. 20. 20. 21. 22. 23. 24. 23.
18. 20. 17. 19. 16. 18. 13. 11.  9. 12. 12.  9.  8.  7.
 7.  8.  5.  6.  6.  6.  7.  6.  8. 12.  8. 11. 13. 12.
 9.  7.  8. 12. 14. 17. 19. 25. 27. 24. 21. 21.  6. 15.
16. 15. 13. 13. 14. 14. 16. 15. 14. 15. 12. 12. 10. 13.
12. 10. 12.  7.  3.  3.  1.  1.  1.  1.  6. 15. 12. 13.
 9.  6.  5.  3.  5.  5.  6.  8.  7.  7.  6.  5.  7.  9.
10. 11. 10.  8.  8.  7. 10. 11. 16. 14. 18. 20. 17. 18.
12. 13. 10. 12. 11. ]
733
```

(3) 做 Linear Regression

```
In [49]: linear_regression(training_x_np, training_y_np, testing_x_np, testing_y_np)
Out[49]: 4.839609070011748
```

(4) 做 Random Forest Regression

```
In [50]: random_forest_regression(training_x_np, training_y_np, testing_x_np, testing_y_np)
Out[50]: 5.560027285129604
```

5. 將未來第一個小時當預測目標， x 取所有 18 種屬性 (e.g. $X[0]$ 會有 18×6 個特徵，即第 0~5 小時的所有 18 種屬性數值)

(1) 這裡整理 training_set 的 x, y 資料

```
In [51]: training_x = new_training_set.iloc[:, 1:]
training_x
```

7	16.70	17.30	21.90	21.50	18.80	17.50	11.80	8.20	10.40	13.20	...	64.10	52.10	48.10	51.80	37.00	21.00	15.40	23.00	16.40
8	18.00	18.00	29.00	29.00	27.00	19.00	23.00	13.00	10.00	13.00	...	30.00	51.00	41.00	38.00	42.00	56.00	56.00	51.00	41.00
9	2.00	4.00	6.00	9.00	10.00	7.00	4.00	6.00	7.00	6.00	...	16.00	35.00	17.00	20.00	22.00	34.00	36.00	32.00	27.00
10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11	89.00	87.00	85.00	84.00	85.00	86.00	88.00	90.00	85.00	75.00	...	54.00	54.00	65.00	75.00	78.00	79.00	81.00	83.00	84.00
12	1.10	1.30	1.40	1.50	1.50	1.60	1.80	2.00	1.90	1.90	...	1.90	4.60	3.90	2.30	3.30	3.50	3.00	3.10	2.40
13	1.65	1.65	1.70	1.73	1.74	1.72	1.78	1.84	1.81	1.81	...	1.83	1.94	1.95	1.94	2.04	2.23	2.20	2.00	2.05
14	292.00	286.00	291.00	286.00	268.00	277.00	247.00	203.00	193.00	200.00	...	281.00	271.00	257.00	246.00	213.00	165.00	198.00	162.00	167.00
15	283.00	289.00	276.00	289.00	250.00	262.00	239.00	183.00	187.00	196.00	...	258.00	273.00	244.00	167.00	178.00	162.00	225.00	170.00	115.00
16	2.10	1.40	2.00	2.50	1.80	2.40	2.00	1.70	2.30	2.80	...	2.10	1.70	1.30	0.80	0.90	1.20	0.80	0.90	0.90
17	0.90	1.10	1.80	2.20	1.80	1.90	2.00	1.00	1.50	2.00	...	1.50	1.60	1.50	1.30	0.60	0.90	0.50	0.60	0.60

18 rows x 1464 columns

```
In [52]: print(training_x.shape[1])
1464
```

```
In [53]: training_x_np = np.zeros((training_x.shape[1] - 6, 18 * 6))
print(len(training_x_np))
print(training_x_np.shape)
1458
(1458, 108)
```

```
In [54]: def get_PM25_Alldata_x_from(data_df, data_np, amount_in_group):
# data_df : 原本的資料
# data_np : 我要存的格式
# amount_in_group : 幾個資料一組
for i in range(data_df.shape[1] - amount_in_group):
    count = 0
    for j in range(18):
        k = i
        for k in range(i, i + 6):
            data_np[i][count] = data_df.iloc[j, k]
            count += 1
            k += 1
    return data_np
training_x_np = get_PM25_Alldata_x_from(training_x, training_x_np, 6)
print(training_x_np)

[[24.7 25.1 25.4 ... 2.2 1.8 1.9]
 [25.1 25.4 25.5 ... 1.8 1.9 2. ]
 [25.4 25.5 25.3 ... 1.9 2. 1. ]
 ...
 [25.1 23.6 22.4 ... 0.6 0.9 0.5]
 [23.6 22.4 22.1 ... 0.9 0.5 0.6]
 [22.4 22.1 21.7 ... 0.5 0.6 0.6]]
```

```
In [55]: training_y_np = np.zeros(len(training_x_np))
def get_PM25_Alldata_y_from(data_df, data_np, amount_in_group):
    for i in range(len(data_np)):
        data_np[i] = data_df.iloc[9, i + 6]
    return data_np
training_y_np = get_PM25_Alldata_y_from(training_x, training_y_np, 6)
print(training_y_np)

[ 4.  6.  7. ... 32. 27. 22.]
```

(2) 再處理 testing set 的 x, y

```
In [56]: testing_x = new_testing_set.iloc[:, 1:]
testing_x_np = np.zeros((testing_x.shape[1] - 6, 18 * 6))
testing_x_np = get_PM25_Alldata_x_from(testing_x, testing_x_np, 6)
print(testing_x_np)
print(len(testing_x_np))
```

```
[[20.5 20.1 19.9 ... 0.8 0.5 0.6]
 [20.1 19.9 19.8 ... 0.5 0.6 0.4]
 [19.9 19.8 20.1 ... 0.6 0.4 0.7]
 ...
 [15.9 15.4 15.3 ... 3.5 3.7 3.5]
 [15.4 15.3 15.3 ... 3.7 3.5 3.1]
 [15.3 15.3 15.1 ... 3.5 3.1 3.2]]
738
```

```
In [57]: testing_y_np = np.zeros(len(testing_x_np))
print(len(testing_y_np))
testing_y_np = get_PM25_Alldata_y_from(testing_x, testing_y_np, 6)
print(testing_y_np)
```

```
5. 10. 6. 8. 7. 7. 12. 9. 6. 6. 8. 7. 6. 7.
8. 6. 6. 16. 15. 17. 8. 11. 17. 17. 8. 11. 10. 14.
12. 11. 19. 22. 14. 16. 19. 17. 18. 15. 13. 8. 9. 14.
10. 7. 3. 4. 4. 5. 6. 4. 3. 5. 4. 7. 9. 9.
11. 10. 6. 6. 8. 11. 8. 9. 8. 13. 11. 10. 8. 8.
10. 13. 12. 13. 13. 12. 16. 12. 11. 18. 15. 15. 15. 13.
13. 14. 11. 9. 17. 13. 12. 9. 7. 8. 11. 11. 13. 9.
9. 10. 10. 9. 8. 8. 13. 15. 13. 15. 13. 15. 13. 16.
15. 20. 19. 10. 21. 22. 24. 30. 31. 32. 30. 31. 26. 28.
29. 31. 30. 11. 19. 11. 14. 18. 21. 21. 19. 15. 12. 12.
8. 13. 14. 11. 7. 9. 9. 11. 9. 9. 19. 20. 20. 20.
21. 22. 23. 24. 23. 18. 20. 17. 19. 16. 18. 13. 11. 9.
12. 12. 9. 8. 7. 7. 8. 5. 6. 6. 6. 7. 6. 8.
12. 8. 11. 13. 12. 9. 7. 8. 12. 14. 17. 19. 25. 27.
24. 21. 21. 6. 15. 16. 15. 13. 13. 14. 14. 16. 15. 14.
15. 12. 12. 10. 13. 12. 10. 12. 7. 3. 3. 1. 1. 1.
1. 6. 15. 12. 13. 9. 6. 5. 3. 5. 5. 6. 8. 7.
7. 6. 5. 7. 9. 10. 11. 10. 8. 8. 7. 10. 11. 16.
14. 18. 20. 17. 18. 12. 13. 10. 12. 11. ]
```

(3) 做 Linear Regression

```
In [58]: linear_regression(training_x_np, training_y_np, testing_x_np, testing_y_np)
```

```
Out [58]: 3.7329306223569962
```

(4) 做 Random Forest Regression

```
In [59]: random_forest_regression(training_x_np, training_y_np, testing_x_np, testing_y_np)
```

```
Out [59]: 3.5657181571815717
```

6. 將未來第六個小時當預測目標 X 有所有 18 種屬性 (e.g. X[0]會有 18*6 個特徵，即第 0~5 小時的所有 18 種屬性數值)

(1) 這裡整理 training_set 的 x, y 資料

```
In [60]: training_x = new_training_set.iloc[:, 1:]
training_x
```

7	16.70	17.30	21.90	21.50	18.80	17.50	11.80	8.20	10.40	13.20	...	64.10	52.10	48.10	51.80	37.00	21.00	15.40	23.00	16.40
8	18.00	18.00	29.00	29.00	27.00	19.00	23.00	13.00	10.00	13.00	...	30.00	51.00	41.00	38.00	42.00	56.00	56.00	51.00	41.00
9	2.00	4.00	6.00	9.00	10.00	7.00	4.00	6.00	7.00	6.00	...	16.00	35.00	17.00	20.00	22.00	34.00	36.00	32.00	27.00
10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11	89.00	87.00	85.00	84.00	85.00	86.00	88.00	90.00	85.00	75.00	...	54.00	54.00	65.00	75.00	78.00	79.00	81.00	83.00	84.00
12	1.10	1.30	1.40	1.50	1.50	1.60	1.80	2.00	1.90	1.90	...	1.90	4.60	3.90	2.30	3.30	3.50	3.00	3.10	2.40
13	1.65	1.65	1.70	1.73	1.74	1.72	1.78	1.84	1.81	1.81	...	1.83	1.94	1.95	1.94	2.04	2.23	2.20	2.00	2.05
14	292.00	286.00	291.00	286.00	268.00	277.00	247.00	203.00	193.00	200.00	...	281.00	271.00	257.00	246.00	213.00	165.00	198.00	162.00	167.00
15	283.00	289.00	276.00	289.00	250.00	262.00	239.00	183.00	187.00	196.00	...	258.00	273.00	244.00	167.00	178.00	162.00	225.00	170.00	115.00
16	2.10	1.40	2.00	2.50	1.80	2.40	2.00	1.70	2.30	2.80	...	2.10	1.70	1.30	0.80	0.90	1.20	0.80	0.90	0.90
17	0.90	1.10	1.80	2.20	1.80	1.90	2.00	1.00	1.50	2.00	...	1.50	1.60	1.50	1.30	0.60	0.90	0.50	0.60	0.60

18 rows x 1464 columns

```
In [61]: training_x_np = np.zeros((training_x.shape[1] - 11, 18 * 6))
training_x_np = get_PM25_Alldata_x_from(training_x, training_x_np, 11)
print(training_x_np)
print(len(training_x_np))

[[24.7 25.1 25.4 ... 2.2 1.8 1.9]
 [25.1 25.4 25.5 ... 1.8 1.9 2. ]
 [25.4 25.5 25.3 ... 1.9 2. 1. ]
 ...
 [23.7 24.9 25.1 ... 1.7 1.5 1.6]
 [24.9 25.1 25.1 ... 1.5 1.6 1.5]
 [25.1 25.1 25.2 ... 1.6 1.5 1.3]]
1453
```

```
In [62]: training_y_np = np.zeros(len(training_x_np))
training_y_np = get_PM25_Alldata_y_from(training_x, training_y_np, 11)
print(training_y_np)
print(len(training_y_np))

[ 4.  6.  7. ... 17. 20. 22.]
1453
```

(2)再處理 testing set 的 x, y

```
In [63]: testing_x = new_testing_set.iloc[:, 1:]
testing_x_np = np.zeros((testing_x.shape[1] - 11, 18 * 6))
testing_x_np = get_PM25_Alldata_x_from(testing_x, testing_x_np, 11)
print(testing_x_np)
print(len(testing_x_np))

[[20.5 20.1 19.9 ... 0.8 0.5 0.6]
 [20.1 19.9 19.8 ... 0.5 0.6 0.4]
 [19.9 19.8 20.1 ... 0.6 0.4 0.7]
 ...
 [16.2 16.8 16.9 ... 3.8 3.5 3.6]
 [16.8 16.9 17.1 ... 3.5 3.6 3.4]
 [16.9 17.1 16.3 ... 3.6 3.4 3.6]]
733
```

```
In [64]: testing_y_np = np.zeros(len(testing_x_np))
testing_y_np = get_PM25_Alldata_y_from(testing_x, testing_y_np, 11)
print(testing_y_np)

[ 4.  6.  7. ... 17. 20. 22.]
```

(3)做 Linear Regression

```
In [65]: linear_regression(training_x_np, training_y_np, testing_x_np, testing_y_np)
Out[65]: 3.7604398413161895
```

(4)做 Random Forest Regression

```
In [66]: random_forest_regression(training_x_np, training_y_np, testing_x_np, testing_y_np)
Out[66]: 3.690995907230559
```