0.讀檔 ¶

In [1]:
```python
import pandas as pd

df = pd.read_csv("./新竹_2020.csv",encoding='big5')
df = df.drop(0) #刪除第一列
df
```

Out[1]:

	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	2020/1/1 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	2020/1/1 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	2020/1/1 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	2020/1/1 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	2020/1/1 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	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	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	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	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	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

6588 rows × 27 columns

1. 資料前處理

(a) 取出*10.11.12*月資料

In [2]:
```python
for i in range(1,len(df)+1):
    month = df.loc[i].iat[1].split("/")[1] #Loc取index
    if month not in ["10", "11", "12"]:
        df = df.drop(index=[i])
df
```

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
4933	新竹	2020/10/1 00:00	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	2020/10/1 00:00	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	2020/10/1 00:00	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	2020/10/1 00:00	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	2020/10/1 00:00	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 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	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	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	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	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

(b) 缺失值以及無效值以前後一小時平均值取代 *(*如果前一小時仍有空值，再取更前一小時*)*

In [3]:
```python
#先刪測站、日期
df = df.drop(df.columns[[0,1]], axis=1) #不用用到欄名的drop欄方法
df
```

Out[3]:

	測項	0	1	2	3	4	5	6	7	8	...	14	15	16	17	18	19	20	21	22	23
4933	AMB_TEMP	23.7	23.8	23.8	23.9	23.9	23.8	24.1	24.7	26	...	29.9	29.6	28.7	27.5	26.4	25.7	25.5	25.3	24.9	24.5
4934	CH4	1.97	1.95	1.96	1.96	1.95	1.96	1.97	1.97	1.96	...	1.97	1.98	1.97	2	2.03	2.04	2.05	2.02	2.1	2.14
4935	CO	0.23	0.22	0.21	0.2	0.2	0.22	0.24	0.29	0.27	...	0.29	0.3	0.33	0.38	0.46	0.5	0.45	0.39	0.46	0.45
4936	NMHC	0.06	0.05	0.03	0.03	0.03	0.04	0.04	0.05	0.06	...	0.06	0.07	0.09	0.11	0.13	0.15	0.1	0.07	0.12	0.18
4937	NO	1.2	0.7	0.5	0.7	0.5	0.3	0.7	0.9	1	...	1.3	1	0.9	0.8	0.5	0.9	0.9	0.3	0.7	0.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6584	THC	2.01	2.02	2	2	1.99	2	1.98	2	2.01	...	2.03	2.07	2.07	2.1	2.1	2.07	2.07	2.05	2.04	2.07
6585	WD_HR	54	55	54	53	58	52	52	35	52	...	54	50	52	45	47	42	42	47	45	44
6586	WIND_DIREC	53	52	57	58	49	54	36	33	52	...	48	43	44	33	50	40	46	46	51	38
6587	WIND_SPEED	4.7	4.6	4.7	4.9	4.1	5.3	5.5	5.6	3.8	...	4.5	4.4	4.2	3.8	3.7	4.7	4.5	4.4	3.9	3.9
6588	WS_HR	3.7	3.6	3.6	3.5	3.5	3.3	3.8	3.8	3.4	...	3.7	3.1	3.3	3.1	2.9	3.3	3.1	2.9	2.8	2.6

1656 rows × 25 columns

In [4]:

```
#轉置df成時間序列data frame: df_t
col_names = df.iloc[0:18].T.iloc[0].tolist() #df_t欄名
col_names = [i.strip() for i in col_names] #消除空白
row = [str(x) for x in range(24)]
df_t = df.iloc[0:18].T.loc[row]
df_t.columns = col_names #rename df_t colname

counter = 18
while counter < 1656:
    df_temp = df.iloc[counter:counter+18].T.loc[row]
    df_temp.columns = col_names #rename df_t colname
    df_t = df_t.append(df_temp)
    counter = counter+18

df_t #2208列是對的，24小時*92天=2208筆
```

Out[4]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19	10.6	1.99	0.31	0.08	1.6	10.8	12.4	28.4	31	13	0	61	#	2.07	42	40	4.7	3.3
20	10.8	1.98	0.29	0.09	1.5	9.3	10.8	28.9	27	12	0	61	#	2.07	42	46	4.5	3.1
21	10.9	1.98	0.28	0.07	1.5	8.6	10.2	29.5	26	15	0	60	#	2.05	47	46	4.4	2.9
22	11	1.97	0.26	0.07	1.4	7.7	9.1	29.7	23	18	0	60	#	2.04	45	51	3.9	2.8
23	11	1.99	0.3	0.08	1.4	9.7	11.1	25.8	27	15	0	62	#	2.07	44	38	3.9	2.6

2208 rows × 18 columns

In [5]:

```
#對SO2最後一列補值=6.9
df_t.iloc[2207]["SO2"] = 6.9
df_t.iloc[2192:]
```

Out[5]:

	AMB_TEMP	CH4	CO	NMHC	NO	NO2	NOx	O3	PM10	PM2.5	RAINFALL	RH	SO2	THC	WD_HR	WIND_DIREC	WIND_SPEED	WS_HR
8	8.7	1.94	0.3	0.07	2.9	10.9	13.8	29.1	34	9	0	56	6.9	2.01	52	52	3.8	3.4
9	9.4	1.95	0.32	0.08	3.9	13.1	17	26.1	37	15	0	56	#	2.03	56	47	3.5	3
10	10.6	1.98	0.33	0.11	5.5	12.7	18.1	26.3	47	17	0	55	x	2.09	44	48	3.8	3
11	11.6	1.98	0.32	0.1	6.7	10.8	17.5	28.7	38	17	0	53	#	2.08	45	44	4.7	3.5
12	11.5	1.99	0.3	0.09	5.3	9.1	14.4	29.2	39	15	0	55	x	2.08	43	47	4.8	4.1
13	11	1.98	0.29	0.07	3.8	8	11.7	29.8	40	18	0	59	#	2.05	49	46	5.4	3.8
14	11.2	1.97	0.28	0.06	3.4	7.4	10.8	30.7	32	15	0	58	#	2.03	54	48	4.5	3.7
15	11	1.98	0.3	0.09	3	10.3	13.3	28.3	36	16	0	60	#	2.07	50	43	4.4	3.1
16	10.8	1.98	0.31	0.09	2.8	11.9	14.8	27.1	25	20	0	61	#	2.07	52	44	4.2	3.3
17	10.5	1.99	0.34	0.11	2.3	13.7	16	25.1	34	12	0	62	#	2.1	45	33	3.8	3.1
18	10.5	1.99	0.34	0.11	2.1	13.1	15.2	26.2	30	17	0	62	#	2.1	47	50	3.7	2.9
19	10.6	1.99	0.31	0.08	1.6	10.8	12.4	28.4	31	13	0	61	#	2.07	42	40	4.7	3.3
20	10.8	1.98	0.29	0.09	1.5	9.3	10.8	28.9	27	12	0	61	#	2.07	42	46	4.5	3.1
21	10.9	1.98	0.28	0.07	1.5	8.6	10.2	29.5	26	15	0	60	#	2.05	47	46	4.4	2.9
22	11	1.97	0.26	0.07	1.4	7.7	9.1	29.7	23	18	0	60	#	2.04	45	51	3.9	2.8
23	11	1.99	0.3	0.08	1.4	9.7	11.1	25.8	27	15	0	62	6.9	2.07	44	38	3.9	2.6

In [6]:

```
#開始進行前後平均補值
miss_list = [] #遺失值樣貌
for i in range(0,len(df_t)):
    for j in range(0,18):
        try:
            float(df_t.iloc[i].iat[j])
        except: #有遺失值
            miss_list.append(df_t.iloc[i].iat[j])
            a=0; l=i-1
            while 1==1:#往前一小時找
                try:
                    a=float(df_t.iloc[l].iat[j])
                    break;
                except:
                    l=l-1
            b=0; l=i+1
            while 1==1:#往後一小時找
                try:
                    b=float(df_t.iloc[l].iat[j])
                    break;
                except:
                    l=l+1
            df_t.iloc[i].iat[j] = (a+b)/2 #前後一小時補值


df_t
```

Out[6]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19	10.6	1.99	0.31	0.08	1.6	10.8	12.4	28.4	31	13	0	61	6.9	2.07	42	40	4.7	3.3
20	10.8	1.98	0.29	0.09	1.5	9.3	10.8	28.9	27	12	0	61	6.9	2.07	42	46	4.5	3.1
21	10.9	1.98	0.28	0.07	1.5	8.6	10.2	29.5	26	15	0	60	6.9	2.05	47	46	4.4	2.9
22	11	1.97	0.26	0.07	1.4	7.7	9.1	29.7	23	18	0	60	6.9	2.04	45	51	3.9	2.8
23	11	1.99	0.3	0.08	1.4	9.7	11.1	25.8	27	15	0	62	6.9	2.07	44	38	3.9	2.6

2208 rows × 18 columns

In [8]:
```
#查看遺失值數量
from collections import Counter
Counter(miss_list)
```

Out[8]:
```
Counter({'#                                ': 324,
         '*                                ': 13,
         'x                                ': 746,
         'A                                ': 23})
```

(c) 將資料切割成訓練集(10.11月)以及測試集(12月)

In [9]:
```
train_df = df_t.iloc[0:1464]
test_df = df_t.iloc[1464:]
```

***(d)*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**

In [10]:
```
train_df = train_df.T
test_df = test_df.T
train_df
```

Out[10]:

	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17	18	19	20	21	22	23
AMB_TEMP	23.7	23.8	23.8	23.9	23.9	23.8	24.1	24.7	26	27.2	...	21.6	21.5	20.4	20	20.1	19.9	19.4	18.9	18.9	18.7
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	0.2	0.2	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	0.2	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	1.2	0.7	0.5	0.7	0.5	0.3	0.7	0.9	1	1.8	...	2.5	2.4	2	1.8	1.6	1.6	1.8	1.7	1.6	1.6
NO2	8	6	5.5	5.2	5.3	5.8	8	7.6	6.6	8	...	4.5	5.4	6.6	9	7.5	8.6	6.9	6	4.8	4.1
NOx	9.2	6.7	6.1	5.8	5.8	6.3	8.6	8.5	7.6	9.8	...	6.9	7.7	8.5	10.8	9.1	10.3	8.7	7.8	6.3	5.7
O3	48	50.6	53.1	53	50.5	47.8	44.8	46.6	51.9	55.8	...	42.4	39.7	35.9	32.4	34.5	33.5	35.2	34.9	36.3	37.8
PM10	21	24	28	26	28	22	26	27	29	23	...	23	30	15	14	14	16	11	18	14	18
PM2.5	16	9	11	10	9	15	10	10	10	9	...	6	9	5	3	4	6	7	9	9	5
RAINFALL	0	0	0	0	0	0	0	0	0	0	...	0.2	0.2	0.4	0.4	0.4	0.6	0.4	0.6	0.8	0.6
RH	72	71	72	72	72	72	72	68	61	55	...	60	60	69	72	72	74	78	82	82	82
SO2	2	2.2	2.3	2.6	2.8	3	3.4	2.9	2.5	2.4	...	2.2	2.8	3.4	3.3	2.8	2.5	2	2.4	2.1	2.1
THC	2.03	2	1.99	1.99	1.98	2	2.01	2.02	2.02	2.05	...	1.98	2	1.99	2.03	2.01	2.04	2.02	2.02	2.02	2.01
WD_HR	49	49	52	55	54	54	55	46	48	47	...	53	52	55	51	60	45	36	47	46	39
WIND_DIREC	57	43	49	60	58	47	54	46	57	38	...	59	43	60	56	57	41	30	55	38	41
WIND_SPEED	3.7	2.9	3.3	3	3.2	2.5	2.9	3.1	4.2	4.3	...	5.5	5.4	4.7	5.1	5.6	5.5	5.8	5.2	4.6	4.8
WS_HR	2.5	2.2	2.5	2.5	2.4	2.3	2.1	2.4	3.1	3.4	...	4.6	4.2	3.8	3.8	4.5	4.1	5.3	3.8	3.4	3.9

18 rows × 1464 columns

2. 時間序列

a.預測目標

***1. Y1:* 將未來第一個小時當預測目標**

In [11]:
```
train_df_Y1 = [float(i) for i in train_df.iloc[9].tolist()[6:]]
test_df_Y1 = [float(i) for i in test_df.iloc[9].tolist()[6:]]
```

***2. Y2:* 將未來第六個小時當預測目標**

In [12]:
```
train_df_Y2 = [float(i) for i in train_df.iloc[9].tolist()[11:]]
test_df_Y2 = [float(i) for i in test_df.iloc[9].tolist()[11:]]
```

b. X分別取

1. X1:* 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 [13]:
```
train_df_X1= []
for i in range(0,1458):
    train_df_X1.append([float(i) for i in train_df.iloc[9].tolist()[i:i+6]])

test_df_X1= []
for i in range(0,738):
    test_df_X1.append([float(i) for i in test_df.iloc[9].tolist()[i:i+6]])
```

2. X2:* 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*)

In [14]:
```
train_df_X2= []
for i in range(0,1458):
    temp = []
    for j in range(0,18):
        temp = temp+[float(i) for i in train_df.iloc[j].tolist()[i:i+6]]
    train_df_X2.append(temp)

test_df_X2= []
for i in range(0,738):
    temp = []
    for j in range(0,18):
        temp = temp+[float(i) for i in test_df.iloc[j].tolist()[i:i+6]]
    test_df_X2.append(temp)
```

c. 使用兩種模型 Linear Regression 和 XGBoost 建模

In [15]:
```
import numpy as np
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor

########################Build model##############################
lm_model = LinearRegression()
xgboostModel = XGBRegressor(n_estimators=1000, learning_rate= 0.3)
########################linear regression########################

######################## model=lm ########################
#x=X1, y=Y1, model=lm
lm_model.fit(train_df_X1, train_df_Y1)
yfit1 = lm_model.predict(test_df_X1)

#x=X1, y=Y2, model=lm
lm_model.fit(train_df_X1[:1453], train_df_Y2)
yfit2 = lm_model.predict(test_df_X1[:733])

#x=X2, y=Y1, model=lm
lm_model.fit(train_df_X2, train_df_Y1)
yfit3 = lm_model.predict(test_df_X2)

#x=X2, y=Y2, model=lm
lm_model.fit(train_df_X2[:1453], train_df_Y2)
yfit4 = lm_model.predict(test_df_X2[:733])

######################## model=xgboost ########################
#x=X1, y=Y1, model=xgboost
xgboostModel.fit(np.array(train_df_X1), np.array(train_df_Y1))
yfit5 = xgboostModel.predict(np.array(test_df_X1))

#x=X1, y=Y2, model=xgboost
xgboostModel.fit(np.array(train_df_X1[:1453]), np.array(train_df_Y2))
yfit6 = xgboostModel.predict(np.array(test_df_X1[:733]))

#x=X2, y=Y1, model=xgboost
xgboostModel.fit(np.array(train_df_X2), np.array(train_df_Y1))
yfit7 = xgboostModel.predict(np.array(test_df_X2))

#x=X2, y=Y2, model=xgboost
xgboostModel.fit(np.array(train_df_X2[:1453]), np.array(train_df_Y2))
yfit8 = xgboostModel.predict(np.array(test_df_X2[:733]))
```

***d.* 用測試集資料計算*MAE* (會有*8*個結果， *2*種*X*資料 *2*種*Y*資料 *2*種模型)**

In [16]:
```
print("x=X1, y=Y1, model=lm MAE:",mean_absolute_error(test_df_Y1,yfit1))
print("x=X1, y=Y2, model=lm MAE:",mean_absolute_error(test_df_Y2,yfit2))
print("x=X2, y=Y1, model=lm MAE:",mean_absolute_error(test_df_Y1,yfit3))
print("x=X2, y=Y2, model=lm MAE:",mean_absolute_error(test_df_Y2,yfit4))
print("x=X1, y=Y1, model=xgboost MAE:",mean_absolute_error(test_df_Y1,yfit5))
print("x=X1, y=Y2, model=xgboost MAE:",mean_absolute_error(test_df_Y2,yfit6))
print("x=X2, y=Y1, model=xgboost MAE:",mean_absolute_error(test_df_Y1,yfit7))
print("x=X2, y=Y2, model=xgboost MAE:",mean_absolute_error(test_df_Y2,yfit8))
```

```
x=X1, y=Y1, model=lm MAE: 2.5223536456517683
x=X1, y=Y2, model=lm MAE: 4.579414220758536
x=X2, y=Y1, model=lm MAE: 2.695868162158899
x=X2, y=Y2, model=lm MAE: 6.088203619636551
x=X1, y=Y1, model=xgboost MAE: 3.07825187696674
x=X1, y=Y2, model=xgboost MAE: 5.201886402674294
x=X2, y=Y1, model=xgboost MAE: 2.9770763649688505
x=X2, y=Y2, model=xgboost MAE: 4.67835432738269
```