```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        from pandas import DataFrame, Series
        import matplotlib.pyplot as plt

        #engine 解決utf8的問題
        data = pd.read_csv("新竹_2019.csv", engine='python')
        data
```

Out[1]:

	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

先 import 可能會用到的套件，接著利用 pandas 套件將資料讀取進來， 先看新竹_2019.csv 含有哪些欄位和資料。

```
In [2]: #print(data.columns)
        #刪除欄位標頭的空白
        data.columns = data.columns.str.strip()
        data.columns
```

```
Out[2]: Index(['測站', '日期', '測項', '00', '01', '02', '03', '04', '05', '06', '07', '08',
               '09', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20',
               '21', '22', '23'],
              dtype='object')
```

新竹_2019.csv 的欄位名稱有空白，假設標頭是測站，印出來會是：
"測站　　"，而不是"測站"，所以會透過 strip()來消除欄位名稱的空白。

In [3]:
```
#清除数據左右的空格
newName = data['測站'].str.strip();
data['測站'] = newName;
newName = data['測項'].str.strip();
data['測項'] = newName;
newName = data['日期'].str.strip();
data['日期'] = newName;
newName = data['00'].str.strip();
data['00'] = newName;
newName = data['01'].str.strip();
data['01'] = newName;
newName = data['02'].str.strip();
data['02'] = newName;
newName = data['03'].str.strip();
data['03'] = newName;
newName = data['04'].str.strip();
data['04'] = newName;
newName = data['05'].str.strip();
data['05'] = newName;
newName = data['06'].str.strip();
data['06'] = newName;
newName = data['07'].str.strip();
data['07'] = newName;
newName = data['08'].str.strip();
data['08'] = newName;
newName = data['09'].str.strip();
```

每個 column 底下的值也有跟上述一樣的問題，所以會透過 strip()來消除空白。

In [4]:
```
#取出10.11.12月資料
start_date = "2019/09/30 00:00:00 "
end_date = "2019/12/31 00:00:00 "

after_start_date = data["日期"] >= start_date
before_end_date = data["日期"] <= end_date
between_two_dates = after_start_date & before_end_date
filtered_data = data.loc[between_two_dates]
#print(filtered_data)
```

接著將資料集 10 到 12 月份的資料取出來。

In [5]:
```
#將index作重置
filtered_data.reset_index(drop=True, inplace=True)
filtered_data
```

Out[5]:

	測站	日期	測項	00	01	02	03	04	05	06	...	14	15	16	17	18	19	20	21	22	23
0	新竹	2019/10/01 00:00:00	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
1	新竹	2019/10/01 00:00:00	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
2	新竹	2019/10/01 00:00:00	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
3	新竹	2019/10/01 00:00:00	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
4	新竹	2019/10/01 00:00:00	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1651	新竹	2019/12/31 00:00:00	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
1652	新竹	2019/12/31 00:00:00	WD_HR	42	41	40	41	38	42	41	...	34	40	40	49	48	51	57	50	49	53
1653	新竹	2019/12/31 00:00:00	WIND_DIREC	35	42	36	47	41	35	35	...	26	44	46	35	50	47	64	40	55	52
1654	新竹	2019/12/31 00:00:00	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
1655	新竹	2019/12/31 00:00:00	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

1656 rows × 27 columns

將資料取出後，會發現前面的 index 是從 4 千多開始，所以利用 reset_index 將 index 重置，讓它重 0 開始。

In [6]:
```
new_data = filtered_data
new_data = new_data.drop(['測站', '日期','測項'], axis=1)
#row和column互轉
new_data = new_data.T
for i in range(len(new_data.columns)):
    new_data[i] = new_data[i].str.replace('A', '#')
    new_data[i] = new_data[i].str.replace('NA', '#')
    new_data[i] = new_data[i].str.replace('x', '#')
    new_data[i] = new_data[i].str.replace('*', '#')
for i in range(len(new_data.columns)):
    for j in range(len(new_data[i])):
        if ('#') in new_data[i][j]:
            new_data[i][j] = np.nan
#print(new_data[1250])
for i in range(len(new_data.columns)):
     new_data[i] = new_data[i].astype('float64')
#使用插值法，缺失值由前一個值和後一個值得平均數；interpolate（）假設函數是線性
new_data = new_data.interpolate()
#print(new_data[1250])
```

接著透過 interpolate()的插值法將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）；因為沒發現 NR，故沒特別做處理。

In [7]:
```
new_data_2 =  filtered_data
#先拿掉測站，無助於分析
new_data_2 = new_data_2.drop(['測站'], axis=1)
new_data_2 = new_data_2.drop(['00', '01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12', '13', '14', '15', '1
#進行轉置
new_data = new_data.T
#兩個資料集合併
dataset = pd.concat([new_data_2, new_data], axis=1)
dataset = dataset.T
#將資料切割成訓練集(10.11月)以及測試集(12月)
X = dataset.iloc[:,0:1098]
y = dataset.iloc[:,1098:1656]
y
```

將資料切割成訓練集(10.11 月)以及測試集(12 月)。

In [8]:
```
#製作時序資料
date_rng = pd.date_range(start='10/01/2019', end='12/01/2019', freq='H')
date_rng_2 = pd.date_range(start='12/01/2019', end='01/01/2020', freq='H')
df = pd.DataFrame(date_rng, columns=['date'])
df_2 = pd.DataFrame(date_rng_2, columns=['date'])
df_2['datetime'] = pd.to_datetime(df_2['date'])
df['datetime'] = pd.to_datetime(df['date'])
#df = df.set_index('datetime')
df.drop(['date'], axis=1, inplace=True)
df_2.drop(['date'], axis=1, inplace=True)
df = df.iloc[0:1464]
df_2 = df_2.iloc[0:744]
df_2
```

Out[8]:

	datetime
0	2019-12-01 00:00:00
1	2019-12-01 01:00:00
2	2019-12-01 02:00:00
3	2019-12-01 03:00:00
4	2019-12-01 04:00:00
...	...
739	2019-12-31 19:00:00
740	2019-12-31 20:00:00
741	2019-12-31 21:00:00
742	2019-12-31 22:00:00
743	2019-12-31 23:00:00

744 rows × 1 columns

先產生逐時數據資料。

In [9]:
```
X = X.T
X = X.drop(['日期'], axis=1)
y = y.T
y = y.drop(['日期'], axis=1)
y
```

Out[9]:

	測項	00	01	02	03	04	05	06	07	08	...	14	15	16	17	18	19	20	21	22	23
1098	AMB_TEMP	20.5	20.1	19.9	19.8	20.1	20.1	19.9	19.7	20.8	...	23	22.6	22.3	22	21.7	21.3	21	21	21	20.7
1099	CH4	1.86	1.91	1.89	1.87	1.96	1.86	1.86	1.9	1.81	...	1.73	1.74	1.75	1.78	1.77	1.75	1.77	1.81	1.79	1.79
1100	CO	0.28	0.29	0.24	0.2	0.23	0.22	0.3	0.46	0.39	...	0.23	0.26	0.28	0.36	0.37	0.32	0.29	0.32	0.27	0.29
1101	NMHC	0.08	0.1	0.09	0.09	0.1	0.08	0.11	0.15	0.12	...	0.04	0.05	0.07	0.12	0.12	0.08	0.09	0.12	0.09	0.09
1102	NO	0.6	0.3	0.3	0.6	0.3	0.96	1.62	2.28	2.94	...	0.3	0.5	0.3	0.3	0.3	0.3	0.3	0.3	0.4	0.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1651	THC	1.81	1.79	1.82	1.84	1.84	1.85	1.85	1.88	1.89	...	1.89	1.9	1.89	1.9	1.88	1.84	1.81	1.81	1.81	1.8
1652	WD_HR	42	41	40	41	38	42	41	45	39	...	34	40	40	49	48	51	57	50	49	53
1653	WIND_DIREC	35	42	36	47	41	35	35	46	44	...	26	44	46	35	50	47	64	40	55	52
1654	WIND_SPEED	3.7	3.5	3.4	3.7	4.6	5	4.8	4.5	4.7	...	3.6	5	3.9	4.7	5.1	4.4	4.1	4.4	3.7	4.6
1655	WS_HR	2.6	2.7	2.9	3.6	3.4	3.5	3.5	3.6	3.7	...	3.5	3.6	3.4	3.6	3.5	3.7	3.5	3.1	3.2	3.6

558 rows × 25 columns

In [10]:
```
#將訓練集每18行合併，轉換成維度為(18,61*24)的DataFrame(每個屬性都有61天*24小時共1464筆資料)
temp = pd.DataFrame({'測項':['AMB_TEMP','CH4','CO','NMHC','NO','NO2','NOx','O3','PM10','PM2.5','RAINFALL','RH','SO2','THC','WD_HR
                    })
a = pd.DataFrame()
new_X = pd.DataFrame()
for i in range(0,1097,18):
    a = X.iloc[i:i+18]
    a.reset_index(drop=True, inplace=True)
    a = a.drop(['測項'], axis=1)
    new_X = pd.concat([new_X, a], axis=1)
new_X = new_X.T
new_X.reset_index(drop=True, inplace=True)
#欄(column)代表逐時數據資料
new_X = pd.concat([df, new_X], axis=1)
new_X = new_X.T
new_X = new_X.rename(index={0:'AMB_TEMP', 1:'CH4', 2:'CO', 3:'NMHC', 4:'NO', 5:'NO2', 6:'NOx', 7:'O3', 8:'PM10', 9:'PM2.5', 10:'
new_X
```

```
#處理測試集
y.reset_index(drop=True, inplace=True)
b = pd.DataFrame()
new_y = pd.DataFrame()
for i in range(0,558,18):
    b = y.iloc[i:i+18]
    b.reset_index(drop=True, inplace=True)
    b = b.drop(['測項'], axis=1)
    new_y = pd.concat([new_y, b], axis=1)
new_y = new_y.T
new_y.reset_index(drop=True, inplace=True)
#欄(column)代表逐時數據資料
new_y = pd.concat([df_2, new_y], axis=1)
new_y = new_y.T
new_y = new_y.rename(index={0:'AMB_TEMP', 1:'CH4', 2:'CO', 3:'NMHC', 4:'NO', 5:'NO2', 6:'NOx', 7:'O3', 8:'PM10', 9:'PM2.5', 10:'
new_y
```

In [11]:
```
#將未來第一個小時當預測目標(第一個 PM2.5)
temp = new_X.T['PM2.5']
temp
temp_2 = pd.DataFrame()
temp_3 = pd.DataFrame()
X_train = temp.iloc[0:6]
y_train = temp.iloc[0:7]
for i in range(1,1464):
    if(i == 1458):
         break
    else:
        temp_2 = temp.iloc[i:i+6]
        temp_2.reset_index(drop=True, inplace=True)
        X_train = pd.concat([X_train, temp_2], axis=1)

        temp_3 = temp.iloc[i:i+7]
        temp_3.reset_index(drop=True, inplace=True)
        y_train = pd.concat([y_train, temp_3], axis=1)
X_train = X_train.T
y_train = y_train.T
y_train = y_train.drop([0,1,2,3,4,5], axis = 1)
y_train
```

```
#處理測試集
temp = new_y.T['PM2.5']
temp_2 = pd.DataFrame()
temp_3 = pd.DataFrame()
X_test = temp.iloc[0:6]
y_test = temp.iloc[0:7]
for i in range(1,744):
    if(i == 738):
         break
    else:
        temp_2 = temp.iloc[i:i+6]
        temp_2.reset_index(drop=True, inplace=True)
        X_test = pd.concat([X_test, temp_2], axis=1)

        temp_3 = temp.iloc[i:i+7]
        temp_3.reset_index(drop=True, inplace=True)
        y_test = pd.concat([y_test, temp_3], axis=1)
X_test = X_test.T
y_test = y_test.T
y_test = y_test.drop([0,1,2,3,4,5], axis = 1)
y_test
```

In [12]:
```
#將未來第一個小時當預測目標(第二個:所有18種屬性)
X_all_train = []
y_all_train = []
for i in range(1,len(new_X.T.columns)):
    temp = new_X.T[new_X.T.columns[i]]
    temp_2 = pd.DataFrame()
    temp_3 = pd.DataFrame()
    X_train_2 = temp.iloc[0:6]
    y_train_2 = temp.iloc[0:7]
    for i in range(1,1464):
        if(i == 1458):
            break
        else:
            temp_2 = temp.iloc[i:i+6]
            temp_2.reset_index(drop=True, inplace=True)
            X_train_2 = pd.concat([X_train_2, temp_2], axis=1)

            temp_3 = temp.iloc[i:i+7]
            temp_3.reset_index(drop=True, inplace=True)
            y_train_2 = pd.concat([y_train_2, temp_3], axis=1)
    X_train_2 = X_train_2.T
    y_train_2 = y_train_2.T
    y_train_2 = y_train_2.drop([0,1,2,3,4,5], axis = 1)
    if (i==2):
        X_all_train = X_train_2
        y_all_train = y_train_2
    else:
        X_all_train.append(X_train_2)
        y_all_train.append(y_train_2)
```

```
A = pd.DataFrame()
X_all_new_train = pd.DataFrame()
X_all_new_train = X_all_train[0]
X_all_new_train.reset_index(drop=True, inplace=True)
for i in range(1,len(X_all_train)):
    A = X_all_train[i]
    A.reset_index(drop=True, inplace=True)
    X_all_new_train = pd.concat([X_all_new_train, A], axis=1)
```

In [14]:
```
#將未來第一個小時當預測目標
A = pd.DataFrame()
y_all_new_train = pd.DataFrame()
y_all_new_train = y_all_train[0]
y_all_new_train.reset_index(drop=True, inplace=True)
for i in range(1,len(y_all_train)):
    A = y_all_train[i]
    A.reset_index(drop=True, inplace=True)
    y_all_new_train = pd.concat([y_all_new_train, A], axis=1)
y_all_new_train
```

Out[14]:

	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
0	24.6	1.75	0.22	0.03	1.2	6.1	7.3	11.8	23	4	0	88	1.8	1.78	247	239	2	2
1	24.6	1.76	0.42	0.08	5.3	9.9	15.2	8.2	13	6	0	90	2	1.84	203	183	1.7	1
2	25.1	1.73	0.36	0.08	5.6	7.4	12.9	10.4	10	7	0	85	1.9	1.81	193	187	2.3	1.5
3	26.8	1.73	0.29	0.08	4.8	6	10.8	13.2	13	6	0	75	1.9	1.81	200	196	2.8	2
4	28.5	1.72	0.27	0.08	4.6	6.1	10.7	16.6	15	4	0	67	1.8	1.8	197	194	2.6	1.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1453	21.7	2	0.69	0.23	1.6	33.4	34.9	21	56	34	0	79	3.5	2.23	165	162	1.2	0.9
1454	21.2	1.97	0.71	0.23	2.6	31.8	34.4	15.4	56	36	0	81	3	2.2	198	225	0.8	0.5
1455	21	1.85	0.48	0.15	1	22.3	23.2	23	51	32	0	83	3.1	2	162	170	0.9	0.6
1456	20.9	1.9	0.5	0.15	1.1	22.7	23.7	16.4	41	27	0	84	2.4	2.05	167	115	0.9	0.6

In [15]:
```
#處理測試集
X_all_test = []
y_all_test = []
for i in range(1,len(new_y.T.columns)):
    temp = new_y.T[new_y.T.columns[i]]
    temp_2 = pd.DataFrame()
    temp_3 = pd.DataFrame()
    X_test_2 = temp.iloc[0:6]
    y_test_2 = temp.iloc[0:7]
    for i in range(1,744):
        if(i == 738):
            break
        else:
            temp_2 = temp.iloc[i:i+6]
            temp_2.reset_index(drop=True, inplace=True)
            X_test_2 = pd.concat([X_test_2, temp_2], axis=1)

            temp_3 = temp.iloc[i:i+7]
            temp_3.reset_index(drop=True, inplace=True)
            y_test_2 = pd.concat([y_test_2, temp_3], axis=1)
    X_test_2 = X_test_2.T
    y_test_2 = y_test_2.T
    y_test_2 = y_test_2.drop([0,1,2,3,4,5], axis = 1)
    if (i==2):
        X_all_test = X_test_2
        y_all_test = y_test_2
    else:
        X_all_test.append(X_test_2)
        y_all_test.append(y_test_2)
```

```
A = pd.DataFrame()
X_all_new_test = pd.DataFrame()
X_all_new_test = X_all_test[0]
X_all_new_test.reset_index(drop=True, inplace=True)
for i in range(1,len(X_all_test)):
    A = X_all_test[i]
    A.reset_index(drop=True, inplace=True)
    X_all_new_test = pd.concat([X_all_new_test, A], axis=1)
X_all_new_test

A = pd.DataFrame()
y_all_new_test = pd.DataFrame()
y_all_new_test = y_all_test[0]
y_all_new_test.reset_index(drop=True, inplace=True)
for i in range(1,len(y_all_test)):
    A = y_all_test[i]
    A.reset_index(drop=True, inplace=True)
    y_all_new_test = pd.concat([y_all_new_test, A], axis=1)
y_all_new_test
```

In [16]:

```
#將未來第六個小時當預測目標
temp = new_X.T['PM2.5']
temp
temp_2 = pd.DataFrame()
temp_3 = pd.DataFrame()
next_X_train = temp.iloc[0:6]
next_y_train = temp.iloc[0:11]
for i in range(1,1464):
    if(i == 1453):
         break
    else:
        temp_2 = temp.iloc[i:i+6]
        temp_2.reset_index(drop=True, inplace=True)
        next_X_train = pd.concat([next_X_train, temp_2], axis=1)

        temp_3 = temp.iloc[i:i+11]
        temp_3.reset_index(drop=True, inplace=True)
        next_y_train = pd.concat([next_y_train, temp_3], axis=1)
next_X_train = next_X_train.T
next_y_train = next_y_train.T
next_y_train = next_y_train.drop([0,1,2,3,4,5,6,7,8,9], axis = 1)
next_X_train
```

In [17]:
```
#處理測試集
#將未來第一個小時當預測目標
temp = new_y.T['PM2.5']
temp_2 = pd.DataFrame()
temp_3 = pd.DataFrame()
next_X_test = temp.iloc[0:6]
next_y_test = temp.iloc[0:11]
for i in range(1,744):
    if(i == 733):
         break
    else:
        temp_2 = temp.iloc[i:i+6]
        temp_2.reset_index(drop=True, inplace=True)
        next_X_test = pd.concat([next_X_test, temp_2], axis=1)

        temp_3 = temp.iloc[i:i+11]
        temp_3.reset_index(drop=True, inplace=True)
        next_y_test = pd.concat([next_y_test, temp_3], axis=1)
next_X_test= next_X_test.T
next_y_test = next_y_test.T
next_y_test = next_y_test.drop([0,1,2,3,4,5,6,7,8,9], axis = 1)
next_X_test
```

In [18]:
```
next_X_all_train = []
next_y_all_train = []
for i in range(1,len(new_X.T.columns)):
    temp = new_X.T[new_X.T.columns[i]]
    temp_2 = pd.DataFrame()
    temp_3 = pd.DataFrame()
    next_X_train_2 = temp.iloc[0:6]
    next_y_train_2 = temp.iloc[0:11]
    for i in range(1,1464):
        if(i == 1453):
             break
        else:
            temp_2 = temp.iloc[i:i+6]
            temp_2.reset_index(drop=True, inplace=True)
            next_X_train_2 = pd.concat([next_X_train_2, temp_2], axis=1)

            temp_3 = temp.iloc[i:i+11]
            temp_3.reset_index(drop=True, inplace=True)
            next_y_train_2 = pd.concat([next_y_train_2, temp_3], axis=1)
    next_X_train_2 = next_X_train_2.T
    next_y_train_2 = next_y_train_2.T
    next_y_train_2 = next_y_train_2.drop([0,1,2,3,4,5,6,7,8,9], axis = 1)
    if (i==2):
        next_X_all_train = next_X_train_2
        next_y_all_train = next_y_train_2
    else:
        next_X_all_train.append(next_X_train_2)
        next_y_all_train.append(next_y_train_2)
```

In [19]:
```
A = pd.DataFrame()
next_X_all_new_train = pd.DataFrame()
next_X_all_new_train = next_X_all_train[0]
next_X_all_new_train.reset_index(drop=True, inplace=True)
for i in range(1,len(next_X_all_train)):
    A = next_X_all_train[i]
    A.reset_index(drop=True, inplace=True)
    next_X_all_new_train = pd.concat([next_X_all_new_train, A], axis=1)
next_X_all_new_train
```

In [20]:
```
A = pd.DataFrame()
next_y_all_new_train = pd.DataFrame()
next_y_all_new_train = next_y_all_train[0]
next_y_all_new_train.reset_index(drop=True, inplace=True)
for i in range(1,len(next_y_all_train)):
    A = next_y_all_train[i]
    A.reset_index(drop=True, inplace=True)
    next_y_all_new_train = pd.concat([next_y_all_new_train, A], axis=1)
next_y_all_new_train
```

In [21]:
```
#處理測試集
next_X_all_test = []
next_y_all_test = []
for i in range(1,len(new_y.T.columns)):
    temp = new_y.T[new_y.T.columns[i]]
    temp_2 = pd.DataFrame()
    temp_3 = pd.DataFrame()
    next_X_test_2 = temp.iloc[0:6]
    next_y_test_2 = temp.iloc[0:11]
    for i in range(1,744):
        if(i == 733):
            break
        else:
            temp_2 = temp.iloc[i:i+6]
            temp_2.reset_index(drop=True, inplace=True)
            next_X_test_2 = pd.concat([next_X_test_2, temp_2], axis=1)

            temp_3 = temp.iloc[i:i+11]
            temp_3.reset_index(drop=True, inplace=True)
            next_y_test_2 = pd.concat([next_y_test_2, temp_3], axis=1)
    next_X_test_2 = next_X_test_2.T
    next_y_test_2 = next_y_test_2.T
    next_y_test_2 = next_y_test_2.drop([0,1,2,3,4,5,6,7,8,9], axis = 1)
    if (i==2):
        next_X_all_test = next_X_test_2
        next_y_all_test = next_y_test_2
    else:
        next_X_all_test.append(next_X_test_2)
        next_y_all_test.append(next_y_test_2)
```

In [22]:
```
A = pd.DataFrame()
next_X_all_new_test = pd.DataFrame()
next_X_all_new_test = next_X_all_test[0]
next_X_all_new_test.reset_index(drop=True, inplace=True)
for i in range(1,len(next_X_all_test)):
    A = next_X_all_test[i]
    A.reset_index(drop=True, inplace=True)
    next_X_all_new_test = pd.concat([next_X_all_new_test, A], axis=1)
next_X_all_new_test
```

In [23]:
```
A = pd.DataFrame()
next_y_all_new_test = pd.DataFrame()
next_y_all_new_test = next_y_all_test[0]
next_y_all_new_test.reset_index(drop=True, inplace=True)
for i in range(1,len(next_y_all_test)):
    A = next_y_all_test[i]
    A.reset_index(drop=True, inplace=True)
    next_y_all_new_test = pd.concat([next_y_all_new_test, A], axis=1)
next_y_all_new_test
```

從 In[10]-In[23]都是在處理時間序列的資料，包含訓練資料跟測試資料的處理，先進行將未來第一個小時當預測目標(PM2.5 和 18 種屬性)，再來處理將未來第六個小時當預測目標(PM2.5 和 18 種屬性)。

In [24]:
```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import mean_absolute_error
from sklearn import ensemble, preprocessing, metrics
from sklearn.ensemble import RandomForestClassifier

reg = LinearRegression().fit(X_train,y_train)
y_pred = reg.predict(X_test)
MAE = metrics.mean_absolute_error(y_test, y_pred)

reg_2 = LinearRegression().fit(next_X_train,next_y_train)
y_pred_2 = reg_2.predict(next_X_test)
MAE_2 = metrics.mean_absolute_error(next_y_test, y_pred_2)
```

```
#訓練model
y_train = y_train.astype('int')
X_train = X_train.astype('int')
y_test = y_test.astype('int')
X_test = X_test.astype('int')
Random_forest = RandomForestClassifier(n_estimators = 200)
model = Random_forest.fit(X_train,y_train)
# Predictions
pred = model.predict(X_test)
MAE_3 = metrics.mean_absolute_error(y_test, pred)

next_y_train = next_y_train.astype('int')
next_X_train = next_X_train.astype('int')
next_y_test = next_y_test.astype('int')
next_X_test = next_X_test.astype('int')
Random_forest = ensemble.RandomForestClassifier(n_estimators = 200)
model_2 = Random_forest.fit(next_X_train,next_y_train)
# Predictions
pred_2 = model_2.predict(next_X_test)
MAE_4 = metrics.mean_absolute_error(next_y_test, pred_2)
```

```
#預測將未來第一個小時當預測目標
#PM2.5
print(MAE)
#預測將未來第六個小時當預測目標
#PM2.5
print(MAE_2)
#預測將未來第一個小時當預測目標
#PM2.5
print(MAE_3)
#預測將未來第六個小時當預測目標
#PM2.5
print(MAE_4)
```

```
2.613339309443992
4.570520791503907
3.4390243902439024
5.829467939972715
```

In [25]:
```
X_all_new_train = X_all_new_train.astype('float')
X_all_new_train = X_all_new_train.interpolate()
y_all_new_train = y_all_new_train.astype('float')
y_all_new_train = y_all_new_train.interpolate()
reg_3 = LinearRegression().fit(X_all_new_train,y_all_new_train)
y_pred_3 = reg_3.predict(X_all_new_test)
MAE_5 = metrics.mean_absolute_error(y_all_new_test, y_pred_3)
#預測將未來第一個小時當預測目標
#18個屬性
print(MAE_5)

next_X_all_new_train = next_X_all_new_train.astype('float')
next_X_all_new_train = next_X_all_new_train.interpolate()
next_y_all_new_train = next_y_all_new_train.astype('float')
next_y_all_new_train = next_y_all_new_train.interpolate()
reg_4 = LinearRegression().fit(next_X_all_new_train,next_y_all_new_train)
y_pred_4 = reg_4.predict(next_X_all_new_test)
MAE_6 = metrics.mean_absolute_error(next_y_all_new_test, y_pred_4)
#預測將未來第六個小時當預測目標
#18個屬性
print(MAE_6)
```

```
Random_forest = ensemble.RandomForestClassifier(n_estimators = 200)
model_3 = Random_forest.fit(X_all_new_train,y_all_new_train.astype('int'))
# Predictions
pred_3 = model_3.predict(X_all_new_test)
MAE_7 = metrics.mean_absolute_error(y_all_new_test, pred_3)
#預測將未來第一個小時當預測目標
#18個屬性
print(MAE_7)

Random_forest = ensemble.RandomForestClassifier(n_estimators = 200)
model_4 = Random_forest.fit(next_X_all_new_train,next_y_all_new_train.astype('int'))
# Predictions
pred_4 = model_4.predict(next_X_all_new_test)
MAE_8 = metrics.mean_absolute_error(next_y_all_new_test, pred_4)
#預測將未來第六個小時當預測目標
#18個屬性
print(MAE_8)
```

```
5.030148821132336
9.44052020284445
6.638024314965371
8.018149790308726
```

In[24]-In[25]使用兩種模型 Linear Regression 和 Random Forest Regression 建模，用測試集資料計算 MAE，總共會有 8 個結果。