Time Series Regression

312706034 資管碩二 張榮翔

程式截圖&簡易說明

- 載入資料集
- 去除不包含資料的row (index = 0)
- 更改DataFrame的欄位名稱

```
import pandas as pd
import numpy as np
#read in xls
df = pd.read_excel('新竹_2021.xls')
#drop the index 0 row
df = df.iloc[1:]
#change the column name
df.columns = ['測站', '日期', '測項', '0' , '1' , '2' , '3' , '4' , '5' , '6' , '7' , '8' , '9' , '10' , '11' , '12' , '13' , '14' , '15' , '16' , '17' ,
df
```

✓ 1.0s

	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	2021-01-01 00:00:00	AMB_TEMP	11.1	11.2	11.4	11.5	11.6	11.7	11.9	...	16.6	16.3	15.6	14.8	14.4	14.5	14.7	14.7	14.6	14.4
2	新竹	2021-01-01 00:00:00	CH4	2.01	1.99	2	2.02	2.03	2.02	2.02	...	1.98	1.97	1.97	2	2.02	2.01	2.01	2	1.98	1.98
3	新竹	2021-01-01 00:00:00	CO	0.31	0.28	0.28	0.33	0.32	0.26	0.25	...	0.31	0.29	0.29	0.33	0.34	0.34	0.34	0.29	0.24	0.21
4	新竹	2021-01-01 00:00:00	NMHC	0.1	0.1	0.08	0.09	0.1	0.07	0.07	...	0.06	0.07	0.08	0.12	0.13	0.1	0.1	0.09	0.05	0.06
5	新竹	2021-01-01 00:00:00	NO	1.5	1.4	1.4	1.5	1.4	1.3	1.4	...	3.5	2.6	2.3	2	1.8	1.8	1.8	1.7	1.5	1.4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6566	新竹	2021-12-31 00:00:00	THC	2.24	2.22	2.19	2.17	2.15	2.1	2.1	...	2.07	2.05	2.09	2.1	2.05	2.1	2.15	2.13	2.09	2.05
6567	新竹	2021-12-31 00:00:00	WD_HR	38	51	50	47	53	53	46	...	51	54	48	53	54	53	47	37	42	48
6568	新竹	2021-12-31 00:00:00	WIND_DIREC	37	59	37	50	62	42	41	...	66	45	40	59	57	55	41	36	53	39
6569	新竹	2021-12-31 00:00:00	WIND_SPEED	2.6	2.6	2.3	2.4	3.4	3.2	3.1	...	4.8	3.2	2.8	3.2	2.5	2.2	1.7	2.5	2.3	1.9
6570	新竹	2021-12-31 00:00:00	WS_HR	2.5	2	2	2	2.5	2.6	2.6	...	3.8	3.2	2.9	2.8	2.4	2	1.6	2	2.1	1.7

6570 rows × 27 columns

- 只保留3個月的資料(10-12月)

```
# only keep the 日期 between October 01 and December 31
from datetime import datetime

# Filter the DataFrame with datetime objects
start_date = datetime.strptime('2021-10-01', '%Y-%m-%d')
end_date = datetime.strptime('2021-12-31', '%Y-%m-%d')

df = df[(df['日期'] >= start_date) & (df['日期'] <= end_date)]
df
```

✓ 0.0s

	測站	日期	測項	0	1	2	3	4	5	6	...	14	15	16	17	18	19	20	21	22	23
4915	新竹	2021-10-01 00:00:00	AMB_TEMP	28.3	28.3	27.8	27.8	27.6	27.6	27.7	...	31.6	31.4	30.9	30.5	30.2	29.8	29.4	29.1	28.7	28.2
4916	新竹	2021-10-01 00:00:00	CH4	2.04	2.02	2.12	2.18	2.19	2.24	2.21	...	1.96	1.97	2.01	2.06	2.07	2.05	2.04	2.03	2.08	2.08
4917	新竹	2021-10-01 00:00:00	CO	0.34	0.3	0.3	0.29	0.3	0.33	0.44	...	0.25	0.27	0.32	0.43	0.45	0.45	0.43	0.42	0.43	0.39
4918	新竹	2021-10-01 00:00:00	NMHC	0.17	0.13	0.12	0.14	0.17	0.16	0.18	...	0.04	0.06	0.05	0.17	0.24	0.22	0.16	0.14	0.16	0.14
4919	新竹	2021-10-01 00:00:00	NO	0.9	0.2	0.5	0.4	0.2	0.6	2.2	...	0.5	0.5	0.5	0.3	0.3	0.3	0.3	0.3	0.4	0.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6566	新竹	2021-12-31 00:00:00	THC	2.24	2.22	2.19	2.17	2.15	2.1	2.1	...	2.07	2.05	2.09	2.1	2.05	2.1	2.15	2.13	2.09	2.05
6567	新竹	2021-12-31 00:00:00	WD_HR	38	51	50	47	53	53	46	...	51	54	48	53	54	53	47	37	42	48
6568	新竹	2021-12-31 00:00:00	WIND_DIREC	37	59	37	50	62	42	41	...	66	45	40	59	57	55	41	36	53	39
6569	新竹	2021-12-31 00:00:00	WIND_SPEED	2.6	2.6	2.3	2.4	3.4	3.2	3.1	...	4.8	3.2	2.8	3.2	2.5	2.2	1.7	2.5	2.3	1.9
6570	新竹	2021-12-31 00:00:00	WS_HR	2.5	2	2	2	2.5	2.6	2.6	...	3.8	3.2	2.9	2.8	2.4	2	1.6	2	2.1	1.7

1656 rows × 27 columns

- 將無效的欄位值(#、*、x、A)轉換為NaN

```
# 將無效值（#、*、x、A）替換為 NaN
invalid_values = ['#', '*', 'x', 'A']
df.replace(invalid_values, np.nan, inplace=True)
```

✓ 0.0s

```
d:\Python\lib\site-packages\pandas\core\frame.py:4524: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().replace(
```

- 將所有數據轉換為數字類型（這會將無法轉換的字符串變為 NaN）

```
# 將所有數據轉換為數字類型（這會將無法轉換的字符串變為 NaN）
df.iloc[:, 3:] = df.iloc[:, 3:].apply(pd.to_numeric, errors='coerce')
```

✓ 0.0s

```
d:\Python\lib\site-packages\pandas\core\indexing.py:1754: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, val, pi)
```

- 確認目前包含NaN值的欄位狀況

```
df.isna().sum()
```

✓ 0.0s

```
測站      0
日期      0
測項      0
0       7
1       4
2       6
3      10
4      10
5      10
6       5
7       5
8       6
9       4
10     33
11     86
12     73
13     30
14     30
15     22
16      6
17      6
18      5
19      6
20     14
21      5
22      7
23      3
dtype: int64
```

- 缺失值以及無效值以前後 1 小時平均值取代 (如果前 1 小時仍有空值，再取更前 1 小時)

```python
def fill_missing_values(data):
    # 定義一個遞歸填充函數
    def recursive_fill(s):
        # 填充前後 1 小時的平均值
        s_filled = s.fillna(s.rolling(window=3, min_periods=1, center=True).mean())

        # 檢查是否仍有缺失值
        if s_filled.isna().any():
            # 如果還有 NaN，遞歸調用
            return recursive_fill(s_filled)
        return s_filled

    return data.apply(recursive_fill, axis=0)

# 遍歷測項，填充每個測項的 NaN 值
for measurement in df['測項'].unique():
    # 選擇特定測項的數據
    subset = df[df['測項'] == measurement].iloc[:, 3:]  # 假設前 3 列是 '測站', '日期', '測項'
    # 填充缺失值
    filled_subset = fill_missing_values(subset)
    # 將填充後的數據更新回原 DataFrame
    df.loc[df['測項'] == measurement, df.columns[3:]] = filled_subset
```

✓ 0.4s

```
d:\Python\lib\site-packages\pandas\core\indexing.py:1787: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, val, pi)
```

- 確認目前包含NaN值的欄位狀況(目前已經沒有NaN值)

```python
df.isna().sum()
```

✓ 0.0s

```
測站    0
日期    0
測項    0
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
11    0
12    0
13    0
14    0
15    0
16    0
17    0
18    0
19    0
20    0
21    0
22    0
23    0
dtype: int64
```

- 資料集的切割(10-11月作為訓練集，12月作為測試集)

```
train_start = '2021-10-01'
train_end = '2021-11-30'
test_start = '2021-12-01'
test_end = '2021-12-31'

train_set = df[(df['日期'] >= train_start) & (df['日期'] <= train_end)]
test_set = df[(df['日期'] >= test_start) & (df['日期'] <= test_end)]
```

✓ 0.0s

```
train_set
```

✓ 0.0s

	測站	日期	測項	0	1	2	3	4	5	6	...	14	15	16	17	18	19	20	21	22	23
4915	新竹	2021-10-01	AMB_TEMP	28.30	28.30	27.80	27.80	27.60	27.60	27.70	...	31.60	31.40	30.90	30.50	30.20	29.80	29.40	29.10	28.70	28.20
4916	新竹	2021-10-01	CH4	2.04	2.02	2.12	2.18	2.19	2.24	2.21	...	1.96	1.97	2.01	2.06	2.07	2.05	2.04	2.03	2.08	2.08
4917	新竹	2021-10-01	CO	0.34	0.30	0.30	0.29	0.30	0.33	0.44	...	0.25	0.27	0.32	0.43	0.45	0.45	0.43	0.42	0.43	0.39
4918	新竹	2021-10-01	NMHC	0.17	0.13	0.12	0.14	0.17	0.16	0.18	...	0.04	0.06	0.05	0.17	0.24	0.22	0.16	0.14	0.16	0.14
4919	新竹	2021-10-01	NO	0.90	0.20	0.50	0.40	0.20	0.60	2.20	...	0.50	0.50	0.50	0.30	0.30	0.30	0.30	0.30	0.40	0.60
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6008	新竹	2021-11-30	THC	2.53	2.65	2.40	2.39	2.33	2.07	1.95	...	2.06	2.08	2.12	2.21	2.19	2.21	2.20	2.18	2.15	2.12
6009	新竹	2021-11-30	WD_HR	99.00	146.00	158.00	192.00	244.00	57.00	68.00	...	47.00	45.00	59.00	58.00	53.00	58.00	56.00	47.00	45.00	51.00
6010	新竹	2021-11-30	WIND_DIREC	128.00	249.00	150.00	211.00	248.00	53.00	17.00	...	55.00	55.00	60.00	59.00	52.00	57.00	54.00	35.00	56.00	52.00
6011	新竹	2021-11-30	WIND_SPEED	0.30	0.40	0.60	0.60	0.50	0.90	1.40	...	2.70	3.60	3.20	3.60	2.90	3.80	3.80	4.00	4.40	4.60
6012	新竹	2021-11-30	WS_HR	0.30	0.20	0.50	0.30	0.50	0.30	0.60	...	2.70	2.80	2.90	3.10	2.90	3.40	3.50	3.60	3.80	3.90

1098 rows × 27 columns

```
test_set
```

✓ 0.0s

	測站	日期	測項	0	1	2	3	4	5	6	...	14	15	16	17	18	19	20	21	22	23
6013	新竹	2021-12-01	AMB_TEMP	17.80	17.20	16.70	16.30	15.90	15.50	15.40	...	17.30	17.20	16.90	16.40	16.30	16.30	16.20	16.10	16.10	16.10
6014	新竹	2021-12-01	CH4	2.04	2.04	2.04	2.04	2.03	2.03	2.04	...	2.04	2.03	2.03	2.03	2.05	2.03	2.02	2.02	2.04	2.04
6015	新竹	2021-12-01	CO	0.30	0.29	0.28	0.27	0.26	0.27	0.30	...	0.32	0.31	0.33	0.37	0.43	0.33	0.30	0.30	0.35	0.26
6016	新竹	2021-12-01	NMHC	0.05	0.06	0.04	0.03	0.05	0.05	0.06	...	0.08	0.09	0.10	0.13	0.14	0.10	0.10	0.10	0.12	0.10
6017	新竹	2021-12-01	NO	0.80	0.40	0.40	0.30	0.30	0.30	0.60	...	2.00	2.00	1.60	1.00	1.10	0.90	0.50	0.60	0.50	0.70
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6566	新竹	2021-12-31	THC	2.24	2.22	2.19	2.17	2.15	2.10	2.10	...	2.07	2.05	2.09	2.10	2.05	2.10	2.15	2.13	2.09	2.05
6567	新竹	2021-12-31	WD_HR	38.00	51.00	50.00	47.00	53.00	53.00	46.00	...	51.00	54.00	48.00	53.00	54.00	53.00	47.00	37.00	42.00	48.00
6568	新竹	2021-12-31	WIND_DIREC	37.00	59.00	37.00	50.00	62.00	42.00	41.00	...	66.00	45.00	40.00	59.00	57.00	55.00	41.00	36.00	53.00	39.00
6569	新竹	2021-12-31	WIND_SPEED	2.60	2.60	2.30	2.40	3.40	3.20	3.10	...	4.80	3.20	2.80	3.20	2.50	2.20	1.70	2.50	2.30	1.90
6570	新竹	2021-12-31	WS_HR	2.50	2.00	2.00	2.00	2.50	2.60	2.60	...	3.80	3.20	2.90	2.80	2.40	2.00	1.60	2.00	2.10	1.70

558 rows × 27 columns

- 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

```
# 定義一個函數來處理時序數據
def reshape_time_series(data):
    # 確保數據按日期和測項排序
    data.sort_values(by=['日期', '測項'], inplace=True)

    # 確認數據行數是否正確
    num_attributes = 18  # 18 種屬性
    num_days = data['日期'].nunique()  # 獲取唯一日期數量
    num_hours = 24  # 每天 24 小時

    # 確認行數是否正確
    assert data.shape[0] == num_attributes * num_days, "數據行數不正確"

    # 將數據重塑
    reshaped_data = data.iloc[:, 3:].values.flatten().reshape(num_attributes, num_days * num_hours)

    # 創建新的 DataFrame
    datetime_index = pd.date_range(start=data['日期'].min(), end=data['日期'].max(), freq='H')

    # 設置完整的日期和小時作為列名
    columns = [f"{date.strftime('%Y-%m-%d')} {hour:02d}" for date in pd.date_range(start=data['日期'].min(), end=data['日期'].max()) for hour in range(num_hours)]

    reshaped_df = pd.DataFrame(reshaped_data, index=data['測項'].unique(), columns=columns)

    return reshaped_df
```

✓ 0.0s

```
# 處理訓練集和測試集
train_time_series = reshape_time_series(train_set)
test_time_series = reshape_time_series(test_set)
```

✓ 0.0s

```
<ipython-input-12-e2c666257d2b>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data.sort_values(by=['日期', '測項'], inplace=True)
```

train_time_series

✓ 0.0s

Python

	2021-10-01 00	2021-10-01 01	2021-10-01 02	2021-10-01 03	2021-10-01 04	2021-10-01 05	2021-10-01 06	2021-10-01 07	2021-10-01 08	2021-10-01 09	...	2021-11-30 14	2021-11-30 15	2021-11-30 16	2021-11-30 17	2021-11-30 18	2021-11-30 19	2021-11-30 20	2021-11-30 21	2021-11-30 22	2021-11-30 23
AMB_TEMP	28.30	28.30	27.80	27.80	27.60	27.60	27.70	28.40	30.00	30.90	...	7.50	7.10	8.50	10.00	10.70	12.20	12.10	10.70	8.90	7.20
CH4	18.80	21.10	17.50	20.90	17.00	18.00	10.80	15.40	36.10	60.30	...	2.01	2.07	2.04	2.08	2.15	2.14	2.18	2.22	2.30	2.35
CO	158.00	93.00	118.00	79.00	97.00	81.00	76.00	79.00	63.00	63.00	...	0.15	0.16	0.17	0.17	0.16	0.16	0.14	0.13	0.12	0.11
NMHC	0.04	0.01	0.01	0.03	0.02	0.02	0.01	0.03	0.04	0.06	...	17.00	18.00	26.00	27.00	28.00	25.00	25.00	28.00	30.00	26.00
NO	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	...	6.00	4.90	4.90	4.10	5.50	6.20	4.40	4.20	3.80	3.90
NO2	3.20	3.70	3.50	3.50	3.30	2.60	3.50	3.70	4.30	4.20	...	11.15	12.40	15.50	15.80	14.80	12.30	10.20	9.60	8.90	10.10
NOx	14.90	9.90	7.50	5.80	5.70	5.90	7.50	9.30	10.10	12.20	...	0.30	0.10	0.30	0.60	0.70	0.50	0.30	0.40	0.00	0.00
O3	2.05	2.05	2.03	2.05	2.04	2.06	2.11	2.10	2.10	2.07	...	1.99	1.99	1.98	1.99	2.01	2.04	2.05	2.07	2.07	2.06
PM10	0.21	0.17	0.16	0.14	0.15	0.17	0.23	0.35	0.26	0.28	...	21.00	24.00	17.00	20.00	16.00	21.00	17.00	24.00	20.00	11.00
PM2.5	16.00	15.00	13.00	14.00	12.00	18.00	13.00	15.00	11.00	13.00	...	72.00	52.00	51.00	53.00	76.00	68.00	81.00	52.00	74.00	81.00
RAINFALL	0.90	2.00	1.50	1.20	2.10	0.70	0.90	1.70	1.70	1.00	...	1.30	1.30	1.00	1.10	1.00	1.20	1.00	1.00	0.90	0.90
RH	12.00	13.60	14.10	12.80	12.20	11.80	12.70	12.00	10.10	8.60	...	42.00	43.00	46.00	50.00	53.00	56.00	56.00	57.00	59.00	60.00
SO2	0.80	2.00	1.70	2.20	2.70	1.80	1.80	1.30	1.10	1.30	...	22.30	22.10	21.40	20.80	20.40	20.20	20.00	20.00	19.80	19.80
THC	1.99	2.00	1.99	2.00	2.00	2.00	2.00	2.03	2.04	2.02	...	67.80	70.10	57.50	28.10	25.00	27.90	28.40	16.50	10.90	8.50
WD_HR	14.00	27.00	20.00	22.00	17.00	13.00	22.00	21.00	24.00	18.00	...	38.00	38.00	41.00	40.00	46.00	52.00	50.00	50.00	52.00	75.00
WIND_DIREC	59.00	31.00	45.00	51.00	43.00	68.00	46.00	59.00	43.00	53.00	...	0.11	0.16	0.17	0.19	0.14	0.13	0.10	0.16	0.24	0.23
WIND_SPEED	0.70	0.80	0.60	0.60	0.80	0.60	0.80	1.40	2.60	3.30	...	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
WS_HR	66.00	68.00	69.00	68.00	69.00	77.00	84.00	83.00	77.00	73.00	...	2.70	2.80	2.90	3.10	2.90	3.40	3.50	3.60	3.80	3.90

18 rows × 1464 columns

test_time_series

✓ 0.0s

Python

	2021-12-01 00	2021-12-01 01	2021-12-01 02	2021-12-01 03	2021-12-01 04	2021-12-01 05	2021-12-01 06	2021-12-01 07	2021-12-01 08	2021-12-01 09	...	2021-12-31 14	2021-12-31 15	2021-12-31 16	2021-12-31 17	2021-12-31 18	2021-12-31 19	2021-12-31 20	2021-12-31 21	2021-12-31 22	2021-12-31 23
AMB_TEMP	17.80	17.20	16.70	16.30	15.90	15.50	15.40	15.60	16.50	17.20	...	0.50	0.60	0.60	0.70	0.70	1.40	1.40	1.60	2.00	3.00
CH4	2.06	2.04	2.06	2.03	2.08	2.11	2.15	2.19	2.11	2.09	...	46.70	45.40	44.00	43.30	41.90	41.80	40.60	40.70	39.70	38.10
CO	27.00	26.00	27.00	33.00	29.00	26.00	34.00	33.00	35.00	41.00	...	0.22	0.24	0.23	0.29	0.31	0.29	0.26	0.24	0.28	0.24
NMHC	0.05	0.03	0.03	0.05	0.03	0.04	0.06	0.08	0.08	0.10	...	66.00	56.00	57.00	63.00	55.00	56.00	54.00	56.00	54.00	59.00
NO	3.90	3.80	5.20	4.50	5.40	3.90	4.80	4.70	4.30	5.00	...	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
NO2	65.00	69.00	69.00	71.00	70.00	70.00	70.00	68.00	65.00	62.00	...	7.30	7.50	10.40	12.40	12.70	11.80	12.30	11.10	9.70	8.90
NOx	16.10	22.90	18.80	19.00	16.50	21.30	20.20	11.80	13.40	14.60	...	20.30	20.40	19.90	19.30	19.00	18.70	18.60	18.60	18.70	18.80
O3	2.01	2.01	2.03	2.02	2.04	2.05	2.04	2.03	2.04	2.05	...	2.07	2.05	2.21	2.31	2.27	2.29	2.46	2.71	2.70	2.40
PM10	55.00	66.00	53.00	59.00	64.00	68.00	87.00	79.00	61.00	79.00	...	38.00	24.00	36.00	52.00	60.00	64.00	63.00	52.00	38.00	35.00
PM2.5	28.00	31.00	31.00	33.00	36.00	43.00	28.00	21.00	20.00	28.00	...	0.07	0.06	0.07	0.09	0.10	0.12	0.11	0.07	0.06	0.04
RAINFALL	0.30	0.30	0.40	0.50	0.30	0.60	0.70	0.90	1.50	2.30	...	3.50	4.80	3.50	3.00	3.10	2.20	2.30	1.50	1.20	0.30
RH	3.10	3.20	3.60	3.10	3.00	3.20	3.10	3.40	3.10	3.70	...	93.00	93.00	93.00	93.00	93.00	93.00	94.00	94.00	94.00	94.00
SO2	1.50	0.60	0.70	0.60	0.50	0.50	0.80	1.00	0.80	1.00	...	18.20	17.20	10.60	11.10	13.00	17.30	17.70	12.50	8.90	8.50
THC	36.50	40.90	46.00	45.40	40.80	38.80	38.00	33.00	29.00	21.20	...	2.00	2.01	2.02	2.03	2.04	2.04	2.05	2.04	2.04	2.03
WD_HR	0.32	0.25	0.24	0.24	0.25	0.26	0.27	0.30	0.31	0.31	...	39.00	41.00	40.00	42.00	41.00	49.00	41.00	44.00	41.00	43.00
WIND_DIREC	36.00	31.00	40.00	45.00	34.00	53.00	43.00	54.00	54.00	38.00	...	11.00	10.00	10.00	8.00	9.00	10.00	10.00	10.00	11.00	10.00
WIND_SPEED	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	2.50	2.10	1.80	1.20	0.90	1.00	1.00	1.00	0.90	0.80
WS_HR	8.50	9.30	7.20	7.20	7.20	7.30	9.60	13.80	16.00	15.10	...	3.80	3.20	2.90	2.80	2.40	2.00	1.60	2.00	2.10	1.70

18 rows × 744 columns

- 檢查index的字串，可以發現字串包含許多空格需要移除

```
print(train_time_series.index)
```

✓ 0.0s

```
Index(['AMB_TEMP        ', 'CH4         ', 'CO          ',
       'NMHC        ', 'NO          ', 'NO2         ',
       'NOx         ', 'O3          ', 'PM10        ',
       'PM2.5       ', 'RAINFALL    ', 'RH          ',
       'SO2         ', 'THC         ', 'WD_HR       ',
       'WIND_DIREC  ', 'WIND_SPEED  ', 'WS_HR       '],
      dtype='object')
```

```
print(test_time_series.index)
```

✓ 0.0s

```
Index(['AMB_TEMP        ', 'CH4         ', 'CO          ',
       'NMHC        ', 'NO          ', 'NO2         ',
       'NOx         ', 'O3          ', 'PM10        ',
       'PM2.5       ', 'RAINFALL    ', 'RH          ',
       'SO2         ', 'THC         ', 'WD_HR       ',
       'WIND_DIREC  ', 'WIND_SPEED  ', 'WS_HR       '],
      dtype='object')
```

- 移除index字串中包含的空格

```
# chamge index with strip() method
train_time_series.index = train_time_series.index.str.strip()
test_time_series.index = test_time_series.index.str.strip()
```

✓ 0.0s

```
print(train_time_series.index)
```

✓ 0.0s

```
Index(['AMB_TEMP', 'CH4', 'CO', 'NMHC', 'NO', 'NO2', 'NOx', 'O3', 'PM10',
       'PM2.5', 'RAINFALL', 'RH', 'SO2', 'THC', 'WD_HR', 'WIND_DIREC',
       'WIND_SPEED', 'WS_HR'],
      dtype='object')
```

```
print(test_time_series.index)
```

✓ 0.0s

```
Index(['AMB_TEMP', 'CH4', 'CO', 'NMHC', 'NO', 'NO2', 'NOx', 'O3', 'PM10',
       'PM2.5', 'RAINFALL', 'RH', 'SO2', 'THC', 'WD_HR', 'WIND_DIREC',
       'WIND_SPEED', 'WS_HR'],
      dtype='object')
```

- 準備2種不同的資料集(只有 PM2.5、 所有 18 種屬性)

```
def create_datasets_pm25_only(time_series, forecast_hour):
    X, Y = [], []
    num_points = time_series.shape[1] - forecast_hour - 6

    for i in range(num_points):
        # 只取 PM2.5 的數據
        x_data = time_series.loc['PM2.5'].iloc[i:i + 6].values[:6]  # 取第 0 ~ 5 小時的 PM2.5 數值
        X.append(x_data)

        # 目標 Y: 預測的 PM2.5 值
        y_data = time_series.loc['PM2.5'].iloc[i + 6 + forecast_hour]
        Y.append(y_data)

    return np.array(X), np.array(Y)

def create_datasets_all_features(time_series, forecast_hour):
    X, Y = [], []
    num_points = time_series.shape[1] - forecast_hour - 6

    for i in range(num_points):
        # 取所有 18 種屬性的數據
        x_data = time_series.iloc[:, i:i + 6].values.flatten()  # 取第 0 ~ 5 小時的所有屬性數值
        X.append(x_data)

        # 目標 Y: 預測的 PM2.5 值
        y_data = time_series.loc['PM2.5'].iloc[i + 6 + forecast_hour]
        Y.append(y_data)

    return np.array(X), np.array(Y)
```

✓ 0.0s

- 準備訓練集和測試集的資料

```
# 將未來第 1 個小時當預測目標(訓練集資料)
X_train_pm25, Y_train_pm25 = create_datasets_pm25_only(train_time_series, forecast_hour=0)
X_train_all, Y_train_all = create_datasets_all_features(train_time_series, forecast_hour=0)

# 將未來第 6 個小時當預測目標(訓練集資料)
X_train_pm25_6, Y_train_pm25_6 = create_datasets_pm25_only(train_time_series, forecast_hour=6)
X_train_all_6, Y_train_all_6 = create_datasets_all_features(train_time_series, forecast_hour=6)

# 將未來第 1 個小時當預測目標(測試集資料)
X_test_pm25, Y_test_pm25 = create_datasets_pm25_only(test_time_series, forecast_hour=0)
X_test_all, Y_test_all = create_datasets_all_features(test_time_series, forecast_hour=0)

# 將未來第 6 個小時當預測目標(測試集資料)
X_test_pm25_6, Y_test_pm25_6 = create_datasets_pm25_only(test_time_series, forecast_hour=6)
X_test_all_6, Y_test_all_6 = create_datasets_all_features(test_time_series, forecast_hour=6)

# 訓練集 X, Y 數據
train_sets = [
    (X_train_pm25, Y_train_pm25),      # PM2.5, Target Hour: 1
    (X_train_all, Y_train_all),        # All Features, Target Hour: 1
    (X_train_pm25_6, Y_train_pm25_6), # PM2.5, Target Hour: 6
    (X_train_all_6, Y_train_all_6),    # All Features, Target Hour: 6
]

# 測試集 X, Y 數據
test_sets = [
    (X_test_pm25, Y_test_pm25),        # PM2.5, Target Hour: 1
    (X_test_all, Y_test_all),          # All Features, Target Hour: 1
    (X_test_pm25_6, Y_test_pm25_6),    # PM2.5, Target Hour: 6
    (X_test_all_6, Y_test_all_6),      # All Features, Target Hour: 6
]
```

- 建立訓練用的模型並且import需要的套件

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from xgboost import XGBRegressor


# 設定模型
models = {
    "Linear Regression": LinearRegression(),
    "XGBoost": XGBRegressor(objective='reg:squarederror')
}
```

✓ 1.8s

- 計算2種模型和4種資料組合的MAE表現

```python
# 計算 MAE
results = {}
for (X_train, Y_train), (X_test, Y_test) in zip(train_sets, test_sets):
    for model_name, model in models.items():
        # 訓練模型
        model.fit(X_train, Y_train)
        # 預測
        Y_pred = model.predict(X_test)
        # 計算 MAE
        mae = mean_absolute_error(Y_test, Y_pred)
        results[(model_name, X_train.shape[1], Y_train.shape[0])] = mae
```

✓ 2.3s

- 做特徵名稱的映射並且印出結果

```python
# 特徵名稱映射
feature_names = {
    6: "PM2.5",
    108: "All Features"
}

target_hours = [1, 1, 1, 1, 6, 6, 6, 6]

for i, ((model_name, num_features, _), mae) in enumerate(results.items()):
    target_hour = target_hours[i]  # 從列表中取得對應的預測目標小時
    feature_name = feature_names.get(num_features, f"Unknown ({num_features})")
    print(f"Model: {model_name}, Features: {feature_name}, Target Hour: {target_hour}, MAE: {mae:.4f}")
```

✓ 0.0s

```
Model: Linear Regression, Features: PM2.5, Target Hour: 1, MAE: 8.3753
Model: XGBoost, Features: PM2.5, Target Hour: 1, MAE: 10.8890
Model: Linear Regression, Features: All Features, Target Hour: 1, MAE: 9.4652
Model: XGBoost, Features: All Features, Target Hour: 1, MAE: 12.3722
Model: Linear Regression, Features: PM2.5, Target Hour: 6, MAE: 19.1477
Model: XGBoost, Features: PM2.5, Target Hour: 6, MAE: 18.0462
Model: Linear Regression, Features: All Features, Target Hour: 6, MAE: 21.1730
Model: XGBoost, Features: All Features, Target Hour: 6, MAE: 20.1048
```

- 準備要用來產生上傳到Kaggle的資料

```
# 訓練集 X, Y 數據
train_sets = [
    (X_train_pm25, Y_train_pm25),  # PM2.5, Target Hour: 1
    (X_train_all, Y_train_all),     # All Features, Target Hour: 1
]

# 測試集 X, Y 數據
test_sets = [
    (X_test_pm25, Y_test_pm25),     # PM2.5, Target Hour: 1
    (X_test_all, Y_test_all),       # All Features, Target Hour: 1
]

results = []



for (X_train, Y_train), (X_test, Y_test) in zip(train_sets, test_sets):

    for model_name, model in models.items():
        # 訓練模型
        model.fit(X_train, Y_train)
        # 預測
        Y_pred = model.predict(X_test)

        # 儲存預測結果到結果列表
        results.append({
            'PM2.5': Y_pred
        })

        # 將結果轉換為 DataFrame
        results_df = pd.DataFrame(results)
```

✓ 0.7s

- 查看目前的DataFrame結果

```
results_df
```

✓ 0.0s

	PM2.5
0	[40.8216994927208, 29.347405563458643, 21.7824...
1	[40.88641, 24.362507, 27.874702, 19.10624, 22....
2	[37.92752474465385, 27.8207324537968, 20.24072...
3	[31.577236, 23.66484, 24.185585, 20.169147, 20...

- 將結果處理成可以上傳到Kaggle的格式

```
results_df_0 = results_df['PM2.5'][0]
results_df_0 = {'No':[i for i in range(738)], 'PM2.5': results_df_0}
results_df_0 = pd.DataFrame(results_df_0)
results_df_0.to_csv('submmission_LinearRegression_PM2.5.csv', index=False)
```

✓ 0.1s

```
results_df_1 = results_df['PM2.5'][0]
results_df_1 = {'No':[i for i in range(738)], 'PM2.5': results_df_1}
results_df_1 = pd.DataFrame(results_df_1)
results_df_1.to_csv('submmission_XGBoost_PM2.5.csv', index=False)
```

✓ 0.0s

```
results_df_2 = results_df['PM2.5'][0]
results_df_2 = {'No':[i for i in range(738)], 'PM2.5': results_df_2}
results_df_2 = pd.DataFrame(results_df_2)
results_df_2.to_csv('submmission_LinearRegression_ALL.csv', index=False)
```

✓ 0.0s

```
results_df_3 = results_df['PM2.5'][0]
results_df_3 = {'No':[i for i in range(738)], 'PM2.5': results_df_3}
results_df_3 = pd.DataFrame(results_df_3)
results_df_3.to_csv('submmission_XGBoost_ALL.csv', index=False)
```

✓ 0.0s

- Kaggle上傳結果