Data Preprocessing

導入資料

In [168]:
```
import pandas as pd
import numpy as np
```

In [169]:
```
df = pd.read_excel('新竹_2021.xls')
```

In [170]:
```
df.head()
```

Out[170]:

	測站	日期	測項	0	1	2	3	4	5	6	...	14	15	16	17	18	19	20	21	22	23
0	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	...	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------
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 rows × 27 columns

In [171]:
```
df = df.drop([0])
```

把字串清乾淨，str都會有很多空白接著

In [172]:
```
df.columns = (pd.Series(df.columns).apply(lambda x: x.strip(' ') if isinstance(x,str) else str(x)))
```

In [173]:
```
df.head()
```

Out[173]:

	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

5 rows × 27 columns

In [174]:
```
df['測項'] = df['測項'].apply(lambda x: x.strip())
```

In [175]:
```
feature_name = df['測項'].unique()
feature_name
```

Out[175]:
```
array(['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)
```

In [176]:
```
df.head()
```

Out[176]:

	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

5 rows × 27 columns

篩選日期並調整size

In [177]:
```
df = df.set_index('日期')
```

In [178]:
```
df = df.loc['2021-10-01':'2021-12-31']
df
```

Out[178]:

日期	測站	測項	0	1	2	3	4	5	6	7	...	14	15	16	17	18	19	20	21	22	23
2021-10-01	新竹	AMB_TEMP	28.3	28.3	27.8	27.8	27.6	27.6	27.7	28.4	...	31.6	31.4	30.9	30.5	30.2	29.8	29.4	29.1	28.7	28.2
2021-10-01	新竹	CH4	2.04	2.02	2.12	2.18	2.19	2.24	2.21	2.17	...	1.96	1.97	2.01	2.06	2.07	2.05	2.04	2.03	2.08	2.08
2021-10-01	新竹	CO	0.34	0.3	0.3	0.29	0.3	0.33	0.44	0.62	...	0.25	0.27	0.32	0.43	0.45	0.45	0.43	0.42	0.43	0.39
2021-10-01	新竹	NMHC	0.17	0.13	0.12	0.14	0.17	0.16	0.18	0.23	...	0.04	0.06	0.05	0.17	0.24	0.22	0.16	0.14	0.16	0.14
2021-10-01	新竹	NO	0.9	0.2	0.5	0.4	0.2	0.6	2.2	3.6	...	0.5	0.5	0.5	0.3	0.3	0.3	0.3	0.3	0.4	0.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2021-12-31	新竹	THC	2.24	2.22	2.19	2.17	2.15	2.1	2.1	2.11	...	2.07	2.05	2.09	2.1	2.05	2.1	2.15	2.13	2.09	2.05
2021-12-31	新竹	WD_HR	38	51	50	47	53	53	46	49	...	51	54	48	53	54	53	47	37	42	48
2021-12-31	新竹	WIND_DIREC	37	59	37	50	62	42	41	59	...	66	45	40	59	57	55	41	36	53	39
2021-12-31	新竹	WIND_SPEED	2.6	2.6	2.3	2.4	3.4	3.2	3.1	3	...	4.8	3.2	2.8	3.2	2.5	2.2	1.7	2.5	2.3	1.9
2021-12-31	新竹	WS_HR	2.5	2	2	2	2.5	2.6	2.6	2.5	...	3.8	3.2	2.9	2.8	2.4	2	1.6	2	2.1	1.7

1656 rows × 26 columns

In [179]:
```
data = pd.DataFrame()
for i in feature_name:
    temp_df = df[df['測項']== i]
    #temp_df = temp_df.set_index('日期')
    temp_reshape = pd.DataFrame(np.matrix(temp_df.drop(columns = ['測站','測項'])).reshape(-1))
    data = pd.concat([data,temp_reshape],axis=0)
```

In [180]:
```
data = data.set_index(feature_name)
data = data.apply(pd.to_numeric,errors='coerce')
#data = data.fillna(0)
```

In [181]:
```
#data.to_csv('test.csv')
```

填空

In [182]:
```
nan_indices = np.column_stack([np.where(data.isnull())[0], np.where(data.isnull())[1]])
nan_indices[0]
```

Out[182]:
```
array([   0, 1835], dtype=int64)
```

In [183]:
```
data.iloc[nan_indices[0][0],nan_indices[0][1]]
```

Out[183]:
```
nan
```

In [184]:
```
for i in nan_indices:
    front = 0.0
    behind = 0.0
    #print(i[0],i[1])
    # 找前一個不為空值的
    for x in range(1,len(data.iloc[0])):
        if np.isnan(data.iloc[i[0],i[1]-x]) == False:
            front =  data.iloc[i[0],i[1]-x]
            break;
    # 找後一個不為空值的
    for x in range(1,len(data[0])):
        if np.isnan(data.iloc[i[0],i[1]+x]) == False:
            behind =  data.iloc[i[0],i[1]+x]
            break;
    data.iloc[i[0],i[1]] = (front+behind)/2
    #print('front',front,'behind',behind)
    #print(data.iloc[i[0],i[1]])
```

In [185]:
```
data.head()
```

Out[185]:

	0	1	2	3	4	5	6	7	8	9	...	2198	2199	2200	2201	2202	2203	2204	2205
AMB_TEMP	28.30	28.30	27.80	27.80	27.60	27.60	27.70	28.40	30.00	30.90	...	18.10	17.50	16.80	16.3	16.10	15.80	15.60	15.70
CH4	2.04	2.02	2.12	2.18	2.19	2.24	2.21	2.17	2.13	2.07	...	1.99	1.98	1.99	2.0	1.98	2.01	2.04	2.03
CO	0.34	0.30	0.30	0.29	0.30	0.33	0.44	0.62	0.60	0.45	...	0.29	0.26	0.27	0.3	0.27	0.29	0.33	0.32
NMHC	0.17	0.13	0.12	0.14	0.17	0.16	0.18	0.23	0.20	0.13	...	0.08	0.07	0.10	0.1	0.07	0.09	0.11	0.10
NO	0.90	0.20	0.50	0.40	0.20	0.60	2.20	3.60	2.70	1.30	...	1.90	1.60	1.20	0.8	0.80	0.70	0.80	0.70

5 rows × 2208 columns

In [186]:
```
train_Y = data.T['PM2.5'][:1464]
test_Y = data.T['PM2.5'][1464:].reset_index(drop=True) #避免之後取index麻煩
```

In [187]:
```
train_Y_first = []
test_Y_first = []
train_Y_sixth = []
test_Y_sixth = []
for i in range(len(train_Y)-6):
    train_Y_first.append(train_Y[i+6])
for i in range(len(train_Y)-11):
    train_Y_sixth.append(train_Y[i+11])

for i in range(len(test_Y)-6):
    test_Y_first.append(test_Y[i+6])
for i in range(len(test_Y)-11):
    test_Y_sixth.append(test_Y[i+11])
```

X = PM2.5 Y = First Hour

In [188]:
```
import xgboost as xgb
from sklearn.linear_model import LinearRegression
```

In [189]:
```
# 1. 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)
train_X = data.T['PM2.5'][:1464]
train_X_first = []
for i in range(len(train_X)-6):
    train_X_first.append(train_X[i:i+6])
train_X_first = np.array(train_X_first)
len(train_X_first)
```

Out[189]: 1458

In [190]:
```
test_X = data.T['PM2.5'][1464:]
test_X_first = []
for i in range(len(test_X)-6):
    test_X_first.append(test_X[i:i+6])
test_X_first = np.array(test_X_first)
len(test_X_first)
```

Out[190]: 738

In [191]:
```
#Linear Regression
reg_pm_first = LinearRegression().fit(train_X_first,train_Y_first)
pred_pm_first = reg_pm_first.predict(test_X_first)
#XGBoost
xg_pm_first = xgb.XGBRegressor()
xg_pm_first.fit(train_X_first,train_Y_first)
xg_pred_pm_first = xg_pm_first.predict(test_X_first)
```

X = PM2.5 Y = Sixth Hour

In [192]:
```
train_X = data.T['PM2.5'][:1464]
train_X_sixth = []
for i in range(len(train_X)-11):
    train_X_sixth.append(train_X[i:i+6])
train_X_sixth = np.array(train_X_sixth)
len(train_X_sixth)
```

Out[192]: 1453

In [193]:
```
test_X = data.T['PM2.5'][1464:]
test_X_sixth = []
for i in range(len(test_X)-11):
    test_X_sixth.append(test_X[i:i+6])
test_X_sixth = np.array(test_X_sixth)
len(test_X_sixth)
```

Out[193]: 733

In [194]:
```
#Linear Regression
reg_pm_sixth = LinearRegression().fit(train_X_sixth,train_Y_sixth)
pred_pm_sixth = reg_pm_sixth.predict(test_X_sixth)
#XGBoost
xg_pm_sixth = xgb.XGBRegressor()
xg_pm_sixth.fit(train_X_sixth,train_Y_sixth)
xg_pred_pm_sixth = xg_pm_first.predict(test_X_sixth)
```

X=all Y = First Hour

In [195]:
```
data = data.T
```

In [196]:
```
train_df = data[:1464]
test_df = data[1464:]
```

In [197]:
```
train_df = train_df.T
test_df = test_df.T
print(train_df.shape,test_df.shape)
train_pm = train_df.loc['PM2.5']
test_pm = test_df.loc['PM2.5']
train_pm
```

```
(18, 1464) (18, 744)
```

Out[197]:
```
0       28.0
1       22.0
2       26.0
3       24.0
4       28.0
        ...
1459    36.0
1460    34.0
1461    32.0
1462    32.0
1463    25.0
Name: PM2.5, Length: 1464, dtype: float64
```

In [198]:
```
train_x_all_first=[]

for i in range(len(train_pm)-6):
    temp = np.array(train_df.iloc[:,i:i+6])
    temp = temp.flatten('C')
    train_x_all_first.append(temp)

train_x_all_first = np.array(train_x_all_first)
train_x_all_first.shape
```

Out[198]:
```
(1458, 108)
```

In [199]:
```
test_x_all_first=[]

for i in range(len(test_pm)-6):
    temp = np.array(test_df.iloc[:,i:i+6])
    temp = temp.flatten('C')
    test_x_all_first.append(temp)

test_x_all_first = np.array(test_x_all_first)
test_x_all_first.shape
```

Out[199]:
```
(738, 108)
```

In [200]:
```
#Linear Regression
reg_all_first = LinearRegression().fit(train_x_all_first,train_Y_first)
pred_all_first = reg_all_first.predict(test_x_all_first)
#XGBoost
xg_all_first = xgb.XGBRegressor()
xg_all_first.fit(train_x_all_first,train_Y_first)
xg_pred_all_first = xg_all_first.predict(test_x_all_first)
```

X=All Y=6th

In [201]:
```
train_x_all_sixth=[]

for i in range(len(train_pm)-11):
    temp = np.array(train_df.iloc[:,i:i+6])
    temp = temp.flatten('C')
    train_x_all_sixth.append(temp)

train_x_all_sixth = np.array(train_x_all_sixth)
train_x_all_sixth.shape
```

Out[201]: (1453, 108)

In [202]:
```
test_x_all_sixth=[]

for i in range(len(test_pm)-11):
    temp = np.array(test_df.iloc[:,i:i+6])
    temp = temp.flatten('C')
    test_x_all_sixth.append(temp)

test_x_all_sixth = np.array(test_x_all_sixth)
test_x_all_sixth.shape
```

Out[202]: (733, 108)

In [203]:
```
#Linear Regression
reg_all_sixth = LinearRegression().fit(train_x_all_sixth,train_Y_sixth)
pred_all_sixth = reg_all_sixth.predict(test_x_all_sixth)
#XGBoost
xg_all_sixth = xgb.XGBRegressor()
xg_all_sixth.fit(train_x_all_sixth,train_Y_sixth)
xg_pred_all_sixth = xg_all_sixth.predict(test_x_all_sixth)
```

MAE計算

In [204]:
```
from sklearn.metrics import mean_absolute_error
```

In [205]:
```
print('X=PM2.5, Y=FirstHour, model=lr, MAE =',
      mean_absolute_error(test_Y_first, pred_pm_first))

print('X=PM2.5, Y=SixthHour, model=lr, MAE =',
      mean_absolute_error(test_Y_sixth, pred_pm_sixth))

print('X=ALL, Y=FirstHour, model=lr, MAE =',
      mean_absolute_error(test_Y_first, pred_all_first))

print('X=ALL, Y=SixthHour, model=lr, MAE =',
      mean_absolute_error(test_Y_sixth, pred_all_sixth))

print('X=PM2.5, Y=FirstHour, model=XGB, MAE =',
      mean_absolute_error(test_Y_first, xg_pred_pm_first))

print('X=PM2.5, Y=SixthHour, model=XGB, MAE =',
      mean_absolute_error(test_Y_sixth, xg_pred_pm_sixth))

print('X=ALL, Y=FirstHour, model=XGB, MAE =',
      mean_absolute_error(test_Y_first, xg_pred_all_first))

print('X=ALL, Y=SixthHour, model=XGB, MAE =',
      mean_absolute_error(test_Y_sixth, xg_pred_all_sixth))
```

```
X=PM2.5, Y=FirstHour, model=lr, MAE = 2.67859512930202
X=PM2.5, Y=SixthHour, model=lr, MAE = 4.307073329319158
X=ALL, Y=FirstHour, model=lr, MAE = 2.6472104043328435
X=ALL, Y=SixthHour, model=lr, MAE = 4.257525858305054
X=PM2.5, Y=FirstHour, model=XGB, MAE = 3.103787559561613
X=PM2.5, Y=SixthHour, model=XGB, MAE = 4.741631446805202
X=ALL, Y=FirstHour, model=XGB, MAE = 3.1279468478226082
X=ALL, Y=SixthHour, model=XGB, MAE = 5.205539767934063
```

In [ ]: