Time Series Regression

目錄CONTENTS

01資料前處理

02時間序列

03模型訓練與預測結果

01 資料前處理 - 讀取資料與去掉空格

```
data = pd.read_csv('新竹_2020.csv', encoding='Big5')

data = data[1:] # 一開始第一行是無資料的row

# 去除資料中的空白

def rstrip(input_data):
    return [data.rstrip() for data in input_data]

# 移除columns 及 data中多餘的空格

data.columns = rstrip(data.columns)

for i in range(len(data)):

data.iloc[i] = rstrip(data.iloc[i])

data

data
```

此份資料集有許多多餘的空白,因此在 讀取時,針對欄位與值去掉多餘的空白, 以利後續抓取資料。

	測站	日期	測項	00	01	02	03	04	05	06	 14	15	16	17	18	19	20	21	22	23
1	新竹	2020/01/01 00:00:00	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	2020/01/01 00:00:00	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	2020/01/01 00:00:00	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	2020/01/01 00:00:00	NMHC	0.06	0.07	0.05	0.05	0.05	0.05	0.07	0.09	0.09	0.07	80.0	0.12	0.12	0.16	0.14	0.17	0.2
5	新竹	2020/01/01 00:00:00	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 00:00:00	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 00:00:00	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 00:00:00	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 00:00:00	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 00:00:00	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

01 資料前處理 - 取出10-12月資料

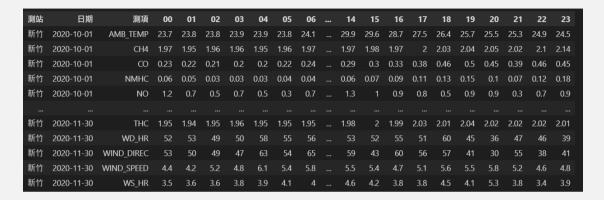
```
1 date = data['日期'] >= '2020/10/01'
2 data = pd.DataFrame(data[date])
3 data['日期'] = pd.to_datetime(data['日期'])
4 data
```

	測站	日期	測項	00	01	02	03	04	05	06	 14	15	16	17	18	19	20	21	22	23
4933	新竹	2020-10-01	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	2020-10-01	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	2020-10-01	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	2020-10-01	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	2020-10-01	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	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	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	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	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	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

01 資料前處理 - 切割訓練集(10、11月)與測試集(12月)

訓練集



	測站	日期	測項	00	01	02	03	04	05	06	14	15	16	17	18	19	20	21	22	23
6031	新竹	2020-12-01	AMB_TEMP	18.5	18.4	18.3	18.2	18.2	18.3	18.4	21.4	20.7	20.1	19.8	19.7	19.9	20.5	20.6	20.5	20.3
6032	新竹	2020-12-01	CH4	1.95	1.95	1.95	1.95	1.95	1.95	1.94	1.92	1.89	1.94	1.95	1.96	1.94	1.91	1.91	1.91	1.92
6033	新竹	2020-12-01	CO	0.17	0.16	0.16	0.16	0.16	0.17	0.17	0.22	0.2	0.25	0.33	0.37	0.32	0.25	0.22	0.21	0.2
6034	新竹	2020-12-01	NMHC	0.06	0.05	0.06	0.06	0.04	0.06	0.03	0.07	0.08	0.11	0.15	0.16	0.12	0.06	0.04	0.04	0.05
6035	新竹	2020-12-01	NO	1.3	1.5	1.2	1.3	1.2	1.3	1.3	#	#	2.3	1.9	1.7	1.6	1.4	1.2	1	1.2
6584	新竹	2020-12-31	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	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	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	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	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
58 row:	s × 27	columns																		

測試集

對資料集做轉置成測項為欄位,以利後續做填補均值,因為接下來模型只需要數值資料,因此把多餘的欄位去除。

4 5 6 7 8 9	2 # 宣告一個新的 DataFrame 來條存合併後的資料 3 train = pd.DataFrame() 4 test = pd.DataFrame() 5 6 7 for head in train_data['測項'].unique(): 8																	
	AMB_TEMP	CH4	со	NMHC	NO	NO2	NOx	О3	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		72	2	2.03	49	57	3.7	2.5
1	23.8	1.95	0.22	0.05	0.7		6.7	50.6	24			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		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		72	2.6	1.99	55	60		2.5
4	23.9	1.95	0.2	0.03	0.5	5.3	5.8	50.5	28	9		72	2.8	1.98	54	58	3.2	2.4
1459	19.9	1.95	0.31	0.09	1.6	8.6	10.3	33.5	16	6	0.6	74	2.5	2.04	45	41	5.5	4.1
1460	19.4	1.95	0.25	0.07	1.8	6.9	8.7	35.2	11		0.4	78	2	2.02	36	30	5.8	5.3
1461	18.9	1.95	0.22	0.07	1.7	6	7.8	34.9	18	9	0.6	82	2.4	2.02	47	55	5.2	3.8
1462	18.9	1.95	0.2	0.07	1.6	4.8	6.3	36.3	14		0.8	82	2.1	2.02	46	38	4.6	3.4
1463	18.7	1.95	0.18	0.06	1.6	4.1	5.7	37.8	18	5	0.6	82	2.1	2.01	39	41	4.8	3.9

訓練集資料

Step 1 - 先將無效字元,統一變為缺失值

```
train.isnull().sum()
 ✓ 0.6s
AMB TEMP
CH4
CO
                6
NMHC
NO
               20
NO2
               20
NOx
               20
03
PM10
PM2.5
RATNFALL
               77
RH
                0
S02
               690
THC
WD HR
WIND DIREC
                1
WIND SPEED
WS_HR
                0
```

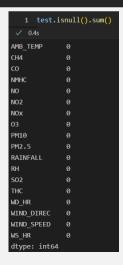
訓練集資料目前缺失值

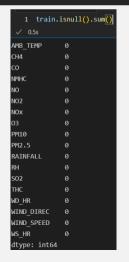
Step 2 – 建立function把原無效字元位置利用前後平均值取代。利用到ffill、bfill方法來取前一值和後一值來算平均。

```
def FillByAvg(df):
    f = pd.DataFrame()
    b = pd.DataFrame()
    for col in df.columns:
        f[col] = df[col].fillna(method='ffill').astype('float')
        b[col] = df[col].fillna(method='bfill').astype('float')
    avg = (f + b) / 2
    return avg
train = FillByAvg(train)
test = FillByAvg(test)
```

Step 3 - 經由觀察還有15個值空白在最後一天的SO2,因為抓取不到後面小時的值(即都為空白值),因此實驗過取當日其他值取中位數、平均與直接填入前一個的值6.9,得出結果皆差不多,那這裡就直接選擇放入6.9,填補這15個的空白。

經由觀察剩下SO2最後一天的最後有15個空白,無法做平均,因此直接填入6.9 test = test.fillna('6.9')





缺失值處理完畢

01 資料前處理 - NR表示無降雨,以0取代

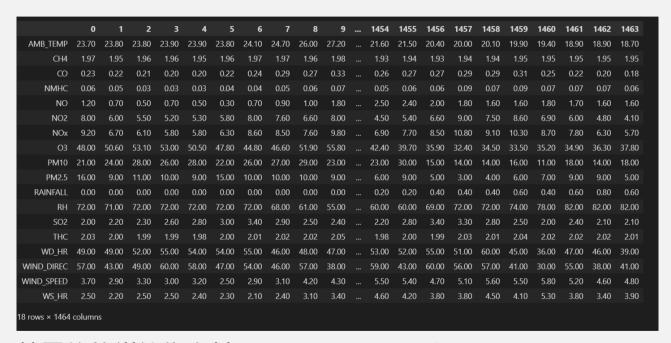
經由觀察,此次資料級,RAINFALL未有NR值,因此不須做處理。

訓練集

```
0.0
       664
0.2
        33
0.8
        11
0.4
        11
0.6
         9
1.0
         6
1.2
         3
2.2
         2
1.4
6.2
         1
2.0
         1
2.6
         1
Name: RAINFALL, dtype: int64
```

測試集 10

01 資料前處理 - 將資料轉換為row代表18種屬性,欄代表數據資料



轉置後的訓練集資料 (18 rows 1464 columns)

02 時間序列 - PM2.5與所有屬性資料準備

將未來第一個小時為預測目標,所以要i + 5 + 1(i + 6)

將未來第六個小時為預測目標·所以要 i + 5 + 6(i+11)

測試集資料與所有屬性資料準備皆與上述大同小異,因此不再放截圖,以免畫面太冗餘。

03 模型訓練與結果 – LinearRegression 調整參數與結果

```
LR pm25 1hr = LinearRegression(fit intercept=False, normalize=True)
LR pm25 1hr.fit(pm25 1hr train X, pm25 1hr train y)
LR pm25 1hr pred = LR pm25 1hr.predict(pm25 1hr test X)
print("MAE of Linear Regression - pm2.5 - 1hr PM2.5: ",
      mean absolute error(pm25 1hr test y, LR pm25 1hr pred))
LR pm25 6hr = LinearRegression(
    fit intercept=False, normalize=True)
LR pm25 6hr.fit(pm25 6hr train X, pm25 6hr train y)
LR pm25 6hr pred = LR pm25 6hr.predict(pm25 6hr test X)
print("MAE of Linear Regression - pm2.5 - 6hr PM2.5: ",
      mean absolute error(pm25 6hr test y, LR pm25 6hr pred))
# Linear Regression - 18 Attributes
LR all 1hr = LinearRegression(
    fit intercept=False, normalize=True)
LR all 1hr.fit(all 1hr train X, all 1hr train y)
LR all 1hr pred = LR all 1hr.predict(all 1hr test X)
print("MAE of Linear Regression - all - 1hr PM2.5: ",
     mean absolute error(all 1hr test y, LR all 1hr pred)
LR all 6hr = LinearRegression(
    fit intercept=False, normalize=True)
LR all 6hr.fit(all 6hr train X, all 6hr train y)
LR all 6hr pred = LR all 6hr.predict(all 6hr test X)
print("MAE of Linear Regression - all - 6hr PM2.5: ",
      mean absolute error(all 6hr test y, LR all 6hr pred))
```

```
MAE of Linear Regression - pm2.5 - 1hr PM2.5: 2.4263326461741257

MAE of Linear Regression - pm2.5 - 6hr PM2.5: 4.349127026907553

MAE of Linear Regression - all - 1hr PM2.5: 2.6958862086920634

MAE of Linear Regression - all - 6hr PM2.5: 5.635087016008428
```

03 模型訓練與結果 - XGBoost 調整參數

XGBoost 參數大多是數值型,因此這裡使用到GridSearchCV來幫忙調整參數

```
5 cv params = {'min child weight': np.linspace(1, 10, 10, dtype=int)}
   6 XGB all 1hr = XGBRegressor(**other params)
     gs = GridSearchCV(XGB all 1hr, cv params, verbose=2,
      gs.fit(all 1hr train X, all 1hr train v)
 10 print("參數的最佳取值: ", gs.best params )
 11 print("最佳模型得分: ", gs.best score )
Fitting 5 folds for each of 10 candidates, totalling 50 fits
參數的最佳取值: {'min child weight': 5}
最佳模型得分: 0.6689032394376048
                                                                                                                  喧 🕨 🗅 日 … 🍵
                                                                                                                     params pm25 1hr = {'eta': 0.3, 'n estimators': 100, 'gamma': 0, 'max depth': 1, 'min child weight': 8,
   1 other params = {'eta': 0.3, 'n estimators': 100, 'gamma': 0, 'max depth': 1, 'min child weight': 5,
   5 cv params = {'gamma': np.linspace(0, 0.1, 12, dtype=int)}
                                                                                                                      params_pm25_6hr = {| 'eta': 0.0774263682681127, 'n_estimators': 100, 'gamma': 0, 'max_depth': 1, 'min_child_weight': 3,
     XGB all 1hr = XGBRegressor(**other params)
                                                                                                                                              'colsample bytree': 1, 'colsample bylevel': 1, 'subsample': 1, 'reg lambda': 60, 'reg alpha': 4,
     gs = GridSearchCV(XGB all 1hr, cv params, verbose=2.
                       refit=True, cv=5, n jobs=-1)
  9 gs.fit(all 1hr train X, all 1hr train y)
                                                                                                                      params all 1hr = {'eta': 0.1291549665014884, 'n estimators': 200, 'gamma': 0, 'max depth': 1, 'min child weight': 1,
 10 print("參數的最佳取值: ", gs.best_params_)
                                                                                                                                      'colsample bytree': 1, 'colsample bylevel': 1, 'subsample': 1, 'reg lambda': 60, 'reg alpha': 4,
 11 print("最佳模型得分: ", gs.best score )
                                                                                                                  14 params all 6hr = {'eta': 0.046415888336127774, 'n estimators': 100, 'gamma': 0, 'max depth': 5, 'min child weight': 1,
itting 5 folds for each of 12 candidates, totalling 60 fits
參數的最佳取值: {'gamma': 0}
                                                                                                                 ✓ 0.4s
最佳模型得分: 0.6689032394376048
```

參數調整畫面

XGBoost不同預測參數調整結果

03 模型訓練與結果 – XGBoost結果

```
XGB pm25 6hr pred = XGB pm25 6hr.predict(pm25 6hr test X)
      print("MAE of XGBoost - pm2.5 - 6hr PM2.5: ",
            mean absolute error(pm25 6hr test v, XGB pm25 6hr pred))
     # XGBoost - 18 Attributes
      XGB all 1hr = XGBRegressor(**params all 1hr)
     XGB all 1hr.fit(all 1hr train X, all 1hr train y)
      XGB all 1hr pred = XGB all 1hr.predict(all 1hr test X)
      print("MAE of XGBoost - all - 1hr PM2.5:",
            mean absolute error(all 1hr test v, XGB all 1hr pred))
      XGB all 6hr = XGBRegressor(**params all 6hr)
      XGB all 6hr.fit(all 6hr train X, all 6hr train y)
      XGB all 6hr pred = XGB all 6hr.predict(all 6hr test X)
      print("MAE of XGBoost - all - 6hr PM2.5: ",
            mean absolute error(all 6hr test y, XGB all 6hr pred))

√ 1.6s

MAE of XGBoost - pm2.5 - 1hr PM2.5: 2.654466922690229
MAE of XGBoost - pm2.5 - 6hr PM2.5: 4.770751389117065
MAE of XGBoost - all - 1hr PM2.5: 2.618156887974519
MAE of XGBoost - all - 6hr PM2.5: 4.436619846427262
```

03 模型訓練與結果 - MAE 表格整理

Linear Regression	預測未來一小時	預測未來六小時
PM2.5	2.43	4.35
所有屬性	2.7	5.64

XGBoost	預測未來一小時	預測未來六小時
PM2.5	2.65	4.77
所有屬性	2.62	4.44