1. 資料前處理

a. 取出10.11.12月資料

```
df = pd.read_excel('107年新竹站_20190315.xls')
df.日期 = pd.to_datetime(df.日期)
data = pd.DataFrame()
for i in range(len(df)):
    if (df['日期'][i].month>9):
        data = pd.concat([data,df.iloc[[i]]])
data = data.reset_index(drop=True)
```

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)

c. NR表示無降雨，以0取代

```
for i in range(len(data)):
    for j in range(3,len(data.columns)):
        if(data.iloc[i,j]=='NR'):
            data.iloc[i,j]= 0    #先把ＮＲ項都先改成0

for i in range(len(data)):
    for j in range(3,len(data.columns)):
        #如果資料含有無效值或空值
        if('#' in str(data.iloc[i,j]) or '*' in str(data.iloc[i,j])
           or 'x' in str(data.iloc[i,j]) or 'A' in str(data.iloc[i,j]) or str(data.iloc[i,j])=='nan'):
            r=i #row
            c=j #column
            #就遞迴往後一筆一筆搜尋(如果後一筆資料不是無效或空值,r跟c的位置就在後一筆，如果後一筆還是無效或空值就繼續往後搜尋)
            while('#' in str(data.iloc[r,c]) or '*' in str(data.iloc[r,c])
               or 'x' in str(data.iloc[r,c]) or 'A' in str(data.iloc[r,c]) or str(data.iloc[r,c])=='nan'):
                if(c==26):#如果是該列最後一位
                    r=r+18 #列數就往後加18，對應到的側項才會是同一個
                    c=3 #該列第一筆資料
                else:
                    c=c+1 #不是該列最後一筆資料的話就直接跑到下一欄
            else:#直到r,c跑到的位置對應資料不再是無效值或空值就執行else(r,c所對應到的資料會是i,j後的下一筆有效資料)
                if(j==3): #如果是該列第一筆資料，就拿18列前的最後一筆資料和現在探索到的下一筆有效資料做平均
                    data.iloc[i,j] = (data.iloc[i-18,26]+data.iloc[r,c])/2
                else: #不是該列第一筆資料的話就直接拿前一欄資料和下一筆有效資料做平均
                    data.iloc[i,j] = (data.iloc[i,j-1]+data.iloc[r,c])/2
```

d. 將資料切割成訓練集(10.11月)以及測試集(12月)

```
#切割10,11月資料做訓練集,12月資料做測試集
train = pd.DataFrame()
test = pd.DataFrame()
for i in range(len(data)):
    if (data['日期'][i].month>11):
        test = pd.concat([test,data.iloc[[i]]])
    else:
        train = pd.concat([train,data.iloc[[i]]])
```

e. 製作時序資料: 將資料形式轉換為欄(column)代表18種屬性，並把column名改成測項名稱以便後續處理資料，行(row)代表逐時數據資料，轉換成維度為(61*24,18)的DataFrame

```
#train資料處理
train.drop('日期',axis=1,inplace=True)
train.drop('測站',axis=1,inplace=True)
measure = train.loc[0:17,['測項']] #把測項資料取出
train.index = train.iloc[:,0] #把train的index改成測項名稱
train.drop('測項',axis=1,inplace=True)
temp = pd.DataFrame()
for i in range(18):
    a = np.array(train.loc[train.index[i],:]) #把同測項名的資料轉成array
    tr = pd.DataFrame(a.reshape(-1)) #把array轉成一維陣列並轉成dataframe(就會是該測項的所有量測數據)
    temp = pd.concat([temp,tr],axis=1) #把全部測項對應的資料合併
train = temp
train.columns = measure.iloc[:,0] #把訓練集的欄位改成測項名稱
```

(截圖為訓練集處理，測試集也是做一樣的處理)

2. 時間序列

a. 取6小時為一單位切割，例如第一筆資料為第0~5小時的資料(X[0])，去預測第6小時的PM2.5值(Y[0])，下一筆資料為第1~6小時的資料(X[1])去預測第7 小時的PM2.5值(Y[1]) *hint: 切割後X的長度應為1464−6=1458

b. X請分別取

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)

```
X_train1 = pd.DataFrame() #只取前六個時間點pm2.5資料作為訓練特徵
for i in range(0,len(train)-6):
    X_train1 = pd.concat([X_train1,train.iloc[i:i+6,9].reset_index(drop=True)],axis=1)
X_train1 = X_train1.T #轉成column為特徵的dataframe
X_train1.reset_index(drop=True,inplace=True) #把index重設
y_train = train['PM2.5'][6:].reset_index(drop=True)
X_test1 = pd.DataFrame() #只取前六個時間點pm2.5資料作為測試特徵
for i in range(0,len(test)-6):
    X_test1 = pd.concat([X_test1,test.iloc[i:i+6,9].reset_index(drop=True)],axis=1)
X_test1 = X_test1.T
X_test1.reset_index(drop=True,inplace=True)
y_test = test['PM2.5'][6:].reset_index(drop=True)
```

2. 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)

```
X_temp = pd.DataFrame() #取前六個時間點18個測項資料作為訓練特徵
for i in range(len(train)): #先把資料降到一維，會是第一個時間點的18個屬性接著第二個時間點的18個屬性，以此類推排列
    X_temp = pd.concat([X_temp,train.iloc[i,:].reset_index(drop=True)],axis=0)
X_temp.reset_index(drop=True,inplace=True)
X_train2 = pd.DataFrame()
for i in range(len(train)-6): #每六個時間點的18個屬性拿出來放到X_train2
    X_train2 = pd.concat([X_train2,X_temp.iloc[i*18:(i*18)+(18*6)].reset_index(drop=True)],axis=1)
X_train2 = X_train2.T #轉成column為特徵的dataframe
X_train2.reset_index(drop=True,inplace=True) #把index重設
X_temp = pd.DataFrame() #取前六個時間點18個測項資料作為測試特徵
for i in range(len(test)):
    X_temp = pd.concat([X_temp,test.iloc[i,:].reset_index(drop=True)],axis=0)
X_temp.reset_index(drop=True,inplace=True)
X_test2 = pd.DataFrame()
for i in range(len(test)-6):
    X_test2 = pd.concat([X_test2,X_temp.iloc[i*18:(i*18)+(18*6)].reset_index(drop=True)],axis=1)
X_test2 = X_test2.T
X_test2.reset_index(drop=True,inplace=True)
y_test = test['PM2.5'][6:].reset_index(drop=True)
```

c. 使用兩種模型 Linear Regression 和 Random Forest Regression 建模

```
lnrg = LinearRegression()
lnrg.fit(X_train1,y_train)
ln_pred1 = lnrg.predict(X_test1)
mae_ln1 = mean_absolute_error(y_test,ln_pred1)
print('MAE (6 features and linear regression) =',mae_ln1)
rfrg = RandomForestRegressor(n_estimators=200, random_state=1234)
rfrg.fit(X_train1,y_train)
rf_pred1 = rfrg.predict(X_test1)
mae_rf1 = mean_absolute_error(y_test,rf_pred1)
print('MAE (6 features and random forest regression) =',mae_rf1)
```

d. 用測試集資料計算MAE (會有4個結果，2種模型*2種X資料)

[**結果**]:

```
MAE (6 features and linear regression) = 2.2365635493180736
MAE (6 features and random forest regression) = 2.7298620144534778
MAE (6*18 features and linear regression) = 2.119799960620856
MAE (6*18 features and random forest regression) = 2.5849762872628728
```

可以推測線性迴歸在時間序列資料預測上表現會比隨機森林迴歸好，在這個情境下使用108個特徵也比使用6個特徵要好