智慧化企業整合 Project3

利用機器學習建立空氣中懸浮粒子預測模型

Individual Research Project

108034532 徐紫芸

目錄

摘要

本研究使用機器學習之方法去預測 $PM_{2.5}$ 濃度，所利用各站空氣品質影響因子數據是由中華民國行政院環境保護署所取得並加以整理，我們將數據分類成兩類：分成訓練資料與測試資料，訓練資料是用於模型構建，測試資料為用於檢測模型構建，只在模型檢驗時使用，用於評估模型的準確率。

初期先以參考文獻找出八個主要因素，並由主要八個因素訓練出最適合的線性模型，再以長短期記憶模型演算法找出比傳統迴歸演算法更精確的結果。

由於新竹為清華大學學生主要活動之地，我們將以新竹偵測站為主，探討各種會影響懸浮粒的生成的因子。其中懸浮粒子影響因子眾多且彼此會互相影響，所以懸浮粒子濃度懸浮微粒的濃度變化並無法使用單因子來解釋其中的相關性，需考慮多個因子共同解釋。

關鍵字：$PM_{2.5}$、機器學習、長短期記憶模型演算法

一、 主題介紹

（一）動機

世界衛生組織所屬的國際癌症研究所於 2013 年宣稱空氣汙染是主要的環境致癌物，而懸浮粒子是戶外空氣污染中的主要成分，會提高膀胱癌的風險以及導致肺癌發生。台灣政府也在 1980 年代開始注重空氣品質問題，於多處設置空氣品質監測站，隨著科技與監測站的快速增加，我們已有足夠的數據能夠探討人為因素能造成的懸浮粒子。

隨著經濟水平提升，台灣人對於環境衛生之品質越發講究，然而重工業發展與道路鋪設所造成的廢氣放量及每年冬季隨著東北季風夾帶大陸內蒙古地區的塵土之自然因素，使得台灣空氣汙染問題日益嚴重。$PM_{2.5}$ 是指一種汙染物，包含固態和液態的顆粒，它的氣動直徑小於 2.5 微米。它會影響肺組織及喉嚨，使哮喘惡化和增加呼吸道疾病。肺癌在台灣近期十大死因排名居高不下，由此可看出其影響程度已危害到民眾的呼吸道健康。身為土生土長的新竹人，對於每天所呼吸的環境應該更加理解，並且能夠事前得知資訊並且加以防範。

（二）目的

1. 空氣品質預測系統可以提供相關的資訊，用來制定較佳的政策，在空氣汙染超過上限值時，可以提出警告。

2. 空氣品質之預測結果，可以協助決策者得到預警，並採取有效的環境管理措施保護民眾的健康。

3. 根據空氣品質之預測結果，高空氣汙染區域之居民和通勤者可以調整其活動來因應。

4. 透過提前而且正確的空氣品質預測，可以為大眾提供有用的資訊，採取預防措施，降低對人體健康之風險。

（三）5W1H

What：預測 $PM_{2.5}$濃度

Why：空氣中的不良物質使人體衍生出許多疾病

Where：新竹地區

When：任何時刻

Who：位在新竹民眾

How：利用機器學習進行預測，建立模型

二、 資料來源

（一） 研究資料來源

1. 中華民國行政院環境保護署所，如下圖所示

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1	日期	測站	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
2	2013/01	新竹	AMB	12	12	12	12	12	13	13	13	13	13	13	13	13	13	14	14	14	14	14	14	14	14	14	15
3	2013/01	新竹	CH4	2	2.1	2	2	2	2	2	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	2	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9
4	2013/01	新竹	CO	0.72	0.7	0.48	0.42	0.43	0.43	0.42	0.44	0.49	0.49	0.52	0.57	0.48	0.51	0.59	0.67	0.53	0.59	0.61	0.52	0.51	0.5	0.42	0.37
5	2013/01	新竹	NMHC	0.3	0.35	0.19	0.21	0.22	0.15	0.13	0.24	0.39	0.2	0.31	0.31	0.26	0.27	0.23	0.21	0.18	0.19	0.19	0.14	0.12	0.11	0.08	0.07
6	2013/01	新竹	NO	10	16	5.3	2.2	3.3	2.5	2.6	3	4.4	7.6	8.6	16	8	8.3	14	20	7.5	7.7	9	5.8	5.5	6.1	2.6	2
7	2013/01	新竹	NO2	30	26	22	21	20	16	16	17	22	22	22	26	19	22	26	27	25	25	24	20	20	20	13	11
8	2013/01	新竹	NOx	41	42	28	24	24	19	19	20	26	29	30	41	27	30	40	47	33	33	33	26	25	26	16	13
9	2013/01	新竹	O3	3.2	0.6	4.9	3.7	3	5.1	7.5	9.5	7.4	7.1	5.9	2.2	5.8	5.3	1.5	0.4	4.9	3.6	2.4	5	4.8	5.9	9.7	9.9
10	2013/01	新竹	PM10	65	73	59	39	42	40	39	39	40	39	33	35	33	27	29	37	34	28	26	21	18	15	13	12
11	2013/01	新竹	PM2.5	49	65	47	30	32	31	34	34	30	31	31	31	34	28	28	33	30	26	23	21	16	13	14	15
12	2013/01	新竹	RAIN	NR	NR	NR	NR	NR	NR	NR	NR	NR	0.2	0.2	0.2	NR	NR	NR	NR	NR	NR	0.2	NR	NR	NR	0.2	NR
13	2013/01	新竹	RH	69	71	72	72	74	76	77	77	79	82	84	85	86	87	87	87	86	85	85	85	84	85	86	85
14	2013/01	新竹	SO2	2.9	2.3	2.3	2	2.8	1.8	3.1	2.5	2.4	1.9	1.4	2.1	1.4	1	1.8	1.9	1.5	1.2	1.2	0.9	1.2	1.6	0.8	1.1
15	2013/01	新竹	THC	2.3	2.5	2.2	2.2	2.2	2.1	2.1	2.2	2.3	2.1	2.2	2.2	2.1	2.2	2.1	2.2	2.1	2.1	2.1	2	2	2	1.9	1.9
16	2013/01	新竹	UVB	0	0	0	0	0	0	0	0.1	0.2	0.5	0.5	0.4	0.6	0.4	0.3	0.1	0	0	0	0	0	0	0	0
17	2013/01	新竹	WD_I	95	187	87	85	90	77	89	83	89	84	85	84	88	84	74	63	83	84	81	83	82	78	81	81
18	2013/01	新竹	WIND	77	155	83	73	90	90	100	92	73	86	82	81	91	82	54	61	83	81	92	81	83	75	82	83
19	2013/01	新竹	WIND	0.6	0.6	0.8	0.5	1.1	0.7	1.6	1.7	1.2	1.9	0.9	1.4	2	1.9	1.4	1.8	1.6	1.8	2	2.2	2	2.5	2.1	1.7

（二） 資料前處理

1. 將所需要的變數拉出

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir
2	1	2013	1	1	0	66	69	14	27	NW	1.3	3.7	0
3	2	2013	1	1	1	61	68	14	23	NW	0.6	3.5	0
4	3	2013	1	1	2	62	72	13	23	NW	0.8	3.1	0
5	4	2013	1	1	3	63	73	13	21	NW	1.3	3.1	0
6	5	2013	1	1	4	60	75	13	20	NW	1.1	3	0
7	6	2013	1	1	5	56	74	13	20	NW	1.5	3.3	0
8	7	2013	1	1	6	55	73	13	24	NW	1.4	3.9	0
9	8	2013	1	1	7	64	74	13	27	NW	2	4.6	0
10	9	2013	1	1	8	62	72	15	29	NW	1.3	4.1	0
11	10	2013	1	1	9	58	64	17	22	NW	1.5	3.7	0
12	11	2013	1	1	10	55	61	18	20	NW	1	3.6	0
13	12	2013	1	1	11	53	58	19	19	NW	1.2	3.5	0
14	13	2013	1	1	12	53	55	20	15	NW	1.2	3.4	0
15	14	2013	1	1	13	51	53	21	12	NW	1.3	3.1	0

2. 讀取資料，並且將日期分類分值。

```
from pandas import read_csv
from datetime import datetime
def parse(x):
    return datetime.strptime(x, '%Y %m %d %H')
dataset = read_csv('raw.csv',  parse_dates = [['year', 'month', 'day', 'hour']], index_col=0, date_parser=parse)
dataset.drop('No', axis=1, inplace=True)
```

3. 在原始資料中，可能會因為某些無法避免的因素導致數據集中有些許的缺失值，這些缺失值會導致機器在學習的過程中有誤差產生。此步驟將資料更改格式，並且處理 NA 空值。

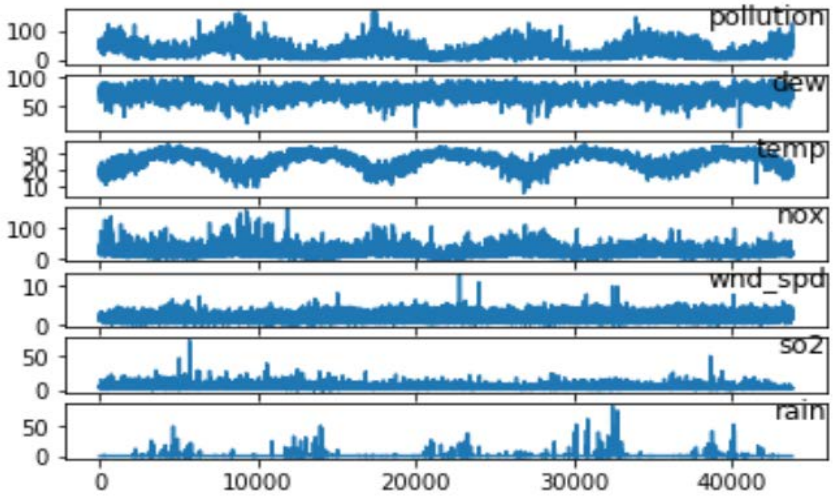
```
dataset.columns = ['pollution', 'dew', 'temp', 'nox', 'wnd_dir', 'wnd_spd', 'so2', 'rain']
dataset.index.name = 'date'
dataset['pollution'].fillna(0, inplace=True)
dataset = dataset[24:]
print(dataset.head(5))
dataset.to_csv('pollution.csv')
```

4. 將處理完成的資料集儲存為 **pollution.csv**

```
dataset.to_csv('pollution.csv')
print("success save as pollution")
```

5. 觀察影響 $PM_{2.5}$ 變因的資料型態。可以發現 $PM_{2.5}$ 濃度與空氣中的化學物以及雨量有密切的關係。

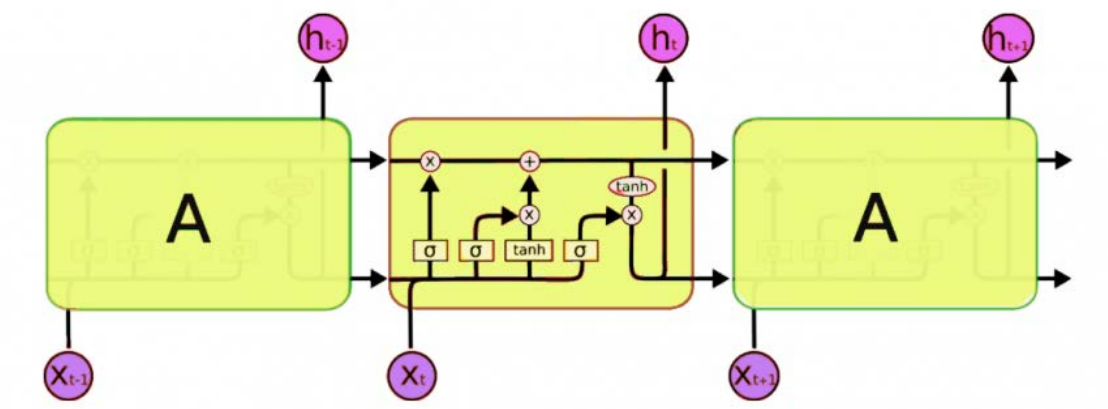
```
dataset = read_csv('pollution.csv', header=0, index_col=0)
values = dataset.values
groups = [0, 1, 2, 3, 5, 6, 7]
i = 1
pyplot.figure()
for group in groups:
    pyplot.subplot(len(groups), 1, i)
    pyplot.plot(values[:, group])
    pyplot.title(dataset.columns[group], y=0.5, loc='right')
    i += 1
pyplot.show()
```



三、 模型架構

（一） LSTM 介紹

長短期記憶 (Long Short-Term Memory, LSTM) 是 RNN 的一種，而其不相同之處在於有了更多的控制單元 input gate、output gate、forget gate 示意圖如下。



（二） 模型建構

在將此研究中所需要的數據及進行處理完成後，便可以開始進行模型的建置。首先，需要設定此模型所需要的神經網路層類別與該神經網路層的層數，以及每層所擁有的神經元個數。將模型的架構堆疊完成後，則是需要針對參數進行選擇，而每個參數代表著不同的意義。Epoch 代表在此訓練過程中，數據被使用了多少次，Keras 中參數更新是按批次進行的，就是小批度下降算法，把數據分為若干組，稱為 Batch，而 Batch_size 則是每組數據的樣本數量。採用 LSTM 模型時，需要對資料進行適配處理，其中包括將資料集轉化為監督學習問題，使模型能夠實現通過前一個時刻（t-1）的汙染資料和天氣條件預測當前時刻（t）的汙染。

```
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
  n_vars = 1 if type(data) is list else data.shape[1]
  df = DataFrame(data)
  cols, names = list(), list()
  for i in range(n_in, 0, -1):
    cols.append(df.shift(i))
    names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
  for i in range(0, n_out):
    cols.append(df.shift(-i))
    if i == 0:
      names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
    else:
      names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
  agg = concat(cols, axis=1)
  agg.columns = names
  if dropnan:
    agg.dropna(inplace=True)
  return agg
```

將處理後的資料集劃分為訓練集和測試集。為了加速模型的訓練，僅利用第一年資料進行訓練，然後利用剩下的 4 年進行評估。

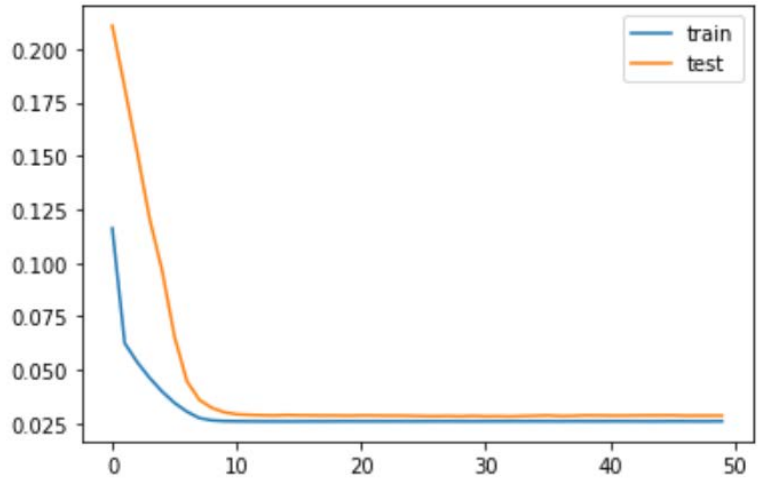
```
values = reframed.values
n_train_hours = 365 * 24
train = values[:n_train_hours, :]
test = values[n_train_hours:, :]
n_obs = n_hours * n_features
train_X, train_y = train[:, :n_obs], train[:, -n_features]
test_X, test_y = test[:, :n_obs], test[:, -n_features]
print(train_X.shape, len(train_X), train_y.shape)
train_X = train_X.reshape((train_X.shape[0], n_hours, n_features))
test_X = test_X.reshape((test_X.shape[0], n_hours, n_features))
print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
```

```
(8760, 1, 8) (8760,) (35039, 1, 8) (35039,)
```

在 LSTM 原始模型中，隱藏層有 50 個神經元，輸出層 1 個神經元，屬於迴歸問題。輸入變數是一個時間步（t-1）的特徵，損失函式採用 Mean Absolute Error(MAE)，優化演算法採用 Adam，模型採用 100 個 epochs，並且每個 batch 的大小為 72。

```
model = Sequential()
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
history = model.fit(train_X, train_y, epochs=100, batch_size=72, validation_data
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```

在 fit( )函式中設定 validation_data 引數，記錄訓練集和測試集的損失，並在完成訓練和測試後繪製損失圖。



最後計算均方根誤差(RMSE)，以及誤差百分比，以作為模型更新的基準。

```
Test RMSE: 6.513
```

```
import numpy

me = numpy.mean( (inv_y - inv_yhat)/inv_yhat,axis = 0)


print(abs(me))
```

```
0.10872321
```

四、 訓練過程

1. 原始模型：

   隱藏層內神經元：50

   Epoch 數：100

   優化演算法：adam

```
model = Sequential()
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
history = model.fit(train_X, train_y, epochs=100, batch_size=72, validation_data
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```

   Test RMSE: 6.513　　誤差率：0.10872321

2. 增加記憶細胞數目：

   隱藏層內神經元：100

   Epoch 數：100

   優化演算法：adam

```
model = Sequential()
model.add(LSTM(100, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
history = model.fit(train_X, train_y, epochs=100, batch_size=72, validation_data
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```

   Test RMSE: 6.507　　誤差率：0.11089778

3. 調整 epoch 數目：

   隱藏層內神經元：50

   Epoch 數：50

   優化演算法：adam

```
model = Sequential()
model.add(LSTM(50, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
history = model.fit(train_X, train_y, epochs=50, batch_size=72, validation_data=
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```

Test RMSE: 6.400　　誤差率：0.07877497

4. 調整 cell 數目：

   隱藏層內神經元：128

   Epoch 數：50

   優化演算法：adam

```
model = Sequential()
model.add(LSTM(128, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
history = model.fit(train_X, train_y, epochs=50, batch_size=72, validation_data=
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```

   Test RMSE: 6.364　　誤差率：0.076035604

5. 調整優化器：

   隱藏層內神經元：128

   Epoch 數：50

   優化演算法：adagrad

```
model = Sequential()
model.add(LSTM(128, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adagrad')
history = model.fit(train_X, train_y, epochs=50, batch_size=72, validation_data=
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```

   Test RMSE: 6.822　　誤差率：0.14410850

6. 增加輸入單位的丟棄率：

隱藏層內神經元：128

Epoch 數：50

優化演算法：adam

丟棄率：0.3

```
model = Sequential()
model.add(LSTM(128, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dropout(0.3))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
history = model.fit(train_X, train_y, epochs=50, batch_size=72, validation_data=
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.legend()
pyplot.show()
```

Test RMSE: 6.196　　誤差率：0.042892274

7. 改善步驟

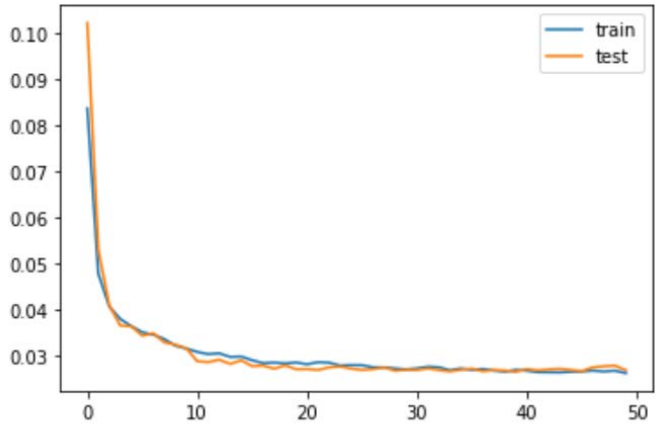
step	神經元數	Epoch 數	優化演算法	丟棄率	誤差百分比
原	50	100	adam	-	10.87%
1	100	100	adam	-	11.09%
2	50	50	adam	-	7.88%
3	128	50	adam	-	7.6%
4	128	50	adagrad	-	14.41%
5	128	50	adam	0.3	4.29%

五、 訓練結果

1. 訓練過程

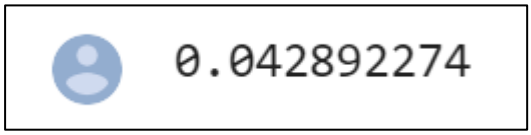
```
Epoch 41/50
 - 2s - loss: 0.0267 - val_loss: 0.0271
Epoch 42/50
 - 2s - loss: 0.0264 - val_loss: 0.0268
Epoch 43/50
 - 2s - loss: 0.0264 - val_loss: 0.0270
Epoch 44/50
 - 2s - loss: 0.0264 - val_loss: 0.0271
Epoch 45/50
 - 2s - loss: 0.0265 - val_loss: 0.0269
Epoch 46/50
 - 2s - loss: 0.0266 - val_loss: 0.0266
Epoch 47/50
 - 2s - loss: 0.0268 - val_loss: 0.0275
Epoch 48/50
 - 2s - loss: 0.0266 - val_loss: 0.0277
Epoch 49/50
 - 2s - loss: 0.0267 - val_loss: 0.0278
Epoch 50/50
 - 2s - loss: 0.0262 - val_loss: 0.0269
```

2. 最佳損失涵數以及均方根誤差




```
Test RMSE: 6.196
```

3. 最低誤差百分比



六、 結果與未來展望

在此研究的實驗數據處理的部分，我們分別 數 據 級 切 割 為 training data、testing data，以及 validation data。在模型訓練中，可以使用 training data 和 validation data 來確認該模型的學習情況，而模型訓練後，可以使用 testing data 進行模型評估及測試，如此便能夠更加了解此預測模型的預測效果，以方便後續修改及優化。未來拓展研究可以結合 CNN 及 LSTM 兩種神經網路，使用 C-LSTM 深度學習模型進行建置，並且另外使用其他迴歸分析的方法進行預測，再將幾種不同的預測方法的預測效果進行比較。此外，此份研究也有一些可以改進的地方，例如影響變因的選擇，應用統計手法尋找最是相關變數；在資料差補的部分可以使用其他方式，例如：最後觀察值法，因為不論是空值或是零值對於預測模型都會有很大的影響。由誤差百分比可以得知，藉由 LSTM 建立具有較高預測能力的預測模型。在最佳化此預測模型後，期望可以建立低誤差空氣汙染預測平台，並將數據視覺化於此平台上，提供給有需求的使用者參考，不僅僅是新竹地區，可以擴及全台，提醒民眾事先做出預防措施，降低 $PM_{2.5}$ 對人體造成的傷害。

七、 文獻探討

[1] Chaloulakou, A., Grivas, G. and Spyrellis, N., 2003, Neural network and multiple regression models for PM10 prediction in Athens: a comparative assessment. *Journal of Air Waste Manage*, 53, 1183–1190.

[2] Mekparyup, J. and Saithanu, K., 2013, Development of Neural Network Technique for Prediction of PM10 Concentration in the Industrial Area, at the East of Thailand, *Applied Mathematical Sciences*, 7(93), 4631-4638.

[3] Pan, B., 2017, Application of XGBoost algorithm in hourly $PM_{2.5}$ concentration prediction, *IOP Conference Series: Earth and Environmental Science*, 113(1), 121-127.

[4] Saeed, S., Hussain, L., Awan, I. A. and Idris, A., 2017, Comparative analysis of different statistical methods for prediction of $PM_{2.5}$ and PM_{10} concentrations in advance for several hours, *International Journal of Computer Science and Network Security*, 17(11), 45-52.

[5] Shahraiyni, H. T. and Sodoudi, S., 2016, Statistical modeling approaches for PM_{10} prediction in urban areas; a review of 21st-century studies, *Atmosphere*, 7(15), 1-24.

[6] Siwek, K. and Osowski, S., 2016, Data mining methods for prediction of air pollution, *International Journal of Applied Mathematics and Computer Science*, 26(2), 467-478.