

IIE Final Project

利用機器學習建立空氣中懸浮粒子預測模型



Presenter : 108034532

徐紫芸



目 錄



PART 01
Scenario/Topic



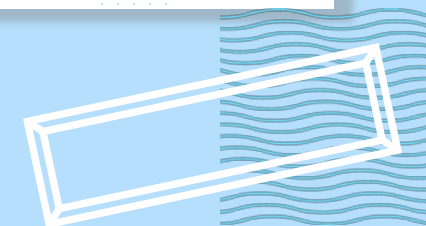
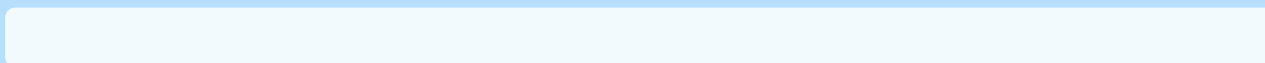
PART 02
Data-preprocessing process



PART 03
Model/Result



PART 04
Conclusion





01 Scenario/Topic

BASIC INFORMATION

Scenario/Topic -5w1h

what

預測PM_{2.5}濃度

when

任何時刻

why

空氣中的不良物質使人體衍生出許多疾病

who

位在新竹民眾

where

新竹地區

how

利用機器學習進行預測，建立模型

Scenario/Topic -target



利用機器學習對PM2.5特性
建立每小時之預測



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。



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。



高空氣汙染區域之居民和通勤者可以調整其活動來因應。



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。



02







Data-preprocessing

Reasonable data-preprocessing process



Data-preprocessing

- 資料來源：中華民國行政院環境保護署所
- 將訓練資料依照年份下載並且合併

 102_HOUR_02_20160225	2019/12/21 下午 03:22	壓縮的 (zipped) ...
 103_HOUR_02_20170317	2019/12/21 下午 03:22	壓縮的 (zipped) ...
 104_HOUR_02_20160323	2019/12/21 下午 03:22	壓縮的 (zipped) ...
 105_HOUR_02_20170301	2019/12/21 下午 03:22	壓縮的 (zipped) ...
 106_HOUR_02_20180308	2019/12/21 下午 03:21	壓縮的 (zipped) ...
 107_HOUR_02_20190315	2019/12/21 下午 03:20	壓縮的 (zipped) ...

Data-preprocessing

資料原形

	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	12	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_F	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

Data-preprocessing

■ 將所需的變數拉出

	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	2010	1	1	0	66	69	14	27	NW	1.3	3.7	0
3	2	2010	1	1	1	61	68	14	23	NW	0.6	3.5	0
4	3	2010	1	1	2	62	72	13	23	NW	0.8	3.1	0
5	4	2010	1	1	3	63	73	13	21	NW	1.3	3.1	0
6	5	2010	1	1	4	60	75	13	20	NW	1.1	3	0
7	6	2010	1	1	5	56	74	13	20	NW	1.5	3.3	0
8	7	2010	1	1	6	55	73	13	24	NW	1.4	3.9	0
9	8	2010	1	1	7	64	74	13	27	NW	2	4.6	0
10	9	2010	1	1	8	62	72	15	29	NW	1.3	4.1	0
11	10	2010	1	1	9	58	64	17	22	NW	1.5	3.7	0
12	11	2010	1	1	10	55	61	18	20	NW	1	3.6	0
13	12	2010	1	1	11	53	58	19	19	NW	1.2	3.5	0
14	13	2010	1	1	12	53	55	20	15	NW	1.2	3.4	0
15	14	2010	1	1	13	51	53	21	12	NW	1.3	3.1	0

Data-preprocessing

■ 讀取資料

```
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)
```

■ 將資料更改格式，並處理空值

```
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')
```

Data-preprocessing

■ pollution.csv

date	pollution	dew	temp	nox	wnd_dir	wnd_spd	so2	rain
2010-01-02 00:00:00	69.0	74	16.0	33.0	SE	1.3	4.2	0.0
2010-01-02 01:00:00	71.0	76	16.0	33.0	SE	1.1	3.0	0.0
2010-01-02 02:00:00	71.0	78	16.0	30.0	SE	1.0	3.0	0.0
2010-01-02 03:00:00	73.0	78	16.0	27.0	SE	0.8	3.3	0.0
2010-01-02 04:00:00	69.0	78	16.0	30.0	SE	1.4	3.2	0.0

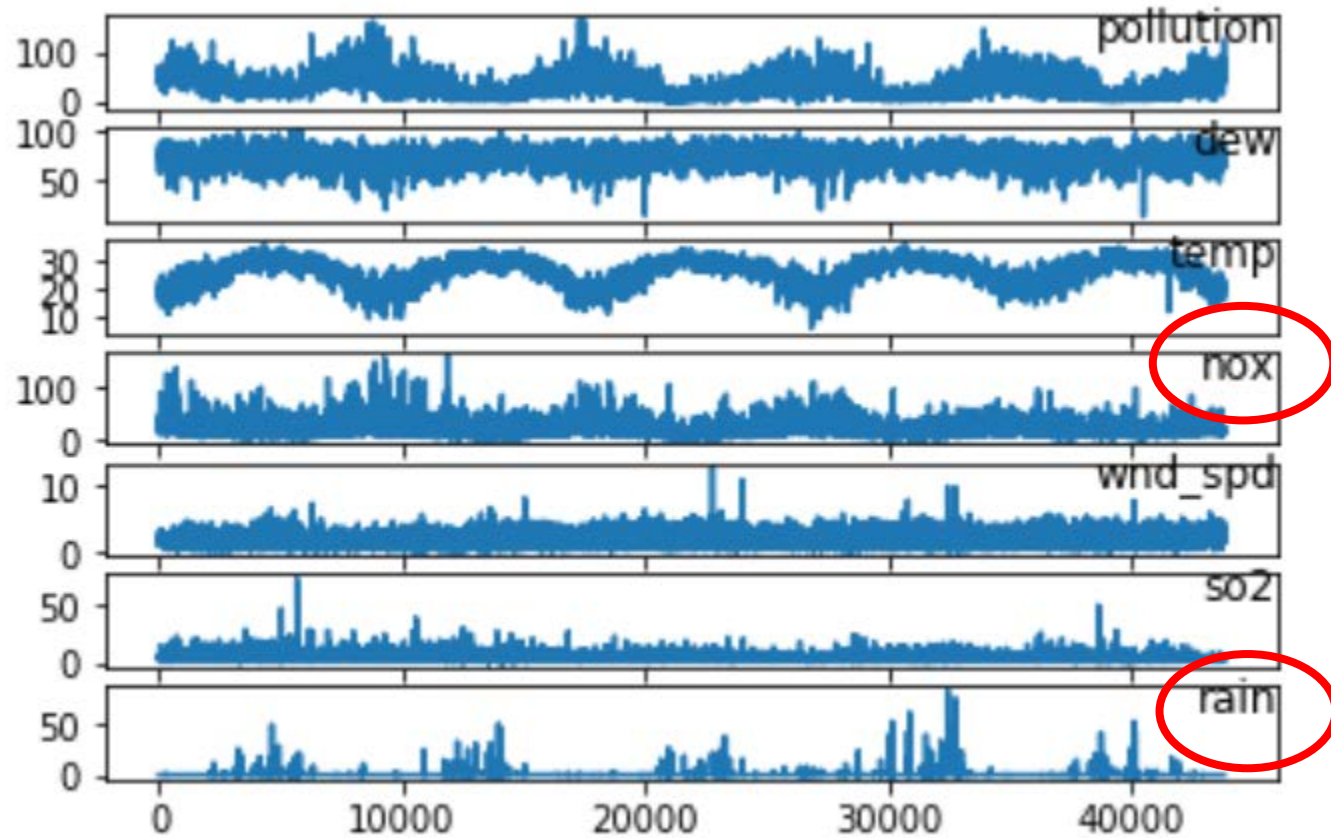
Data-preprocessing

- 觀察影響PM_{2.5}變因的資料型態

```
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()
```

Data-preprocessing

- 觀察影響PM_{2.5}變因的資料型態








03

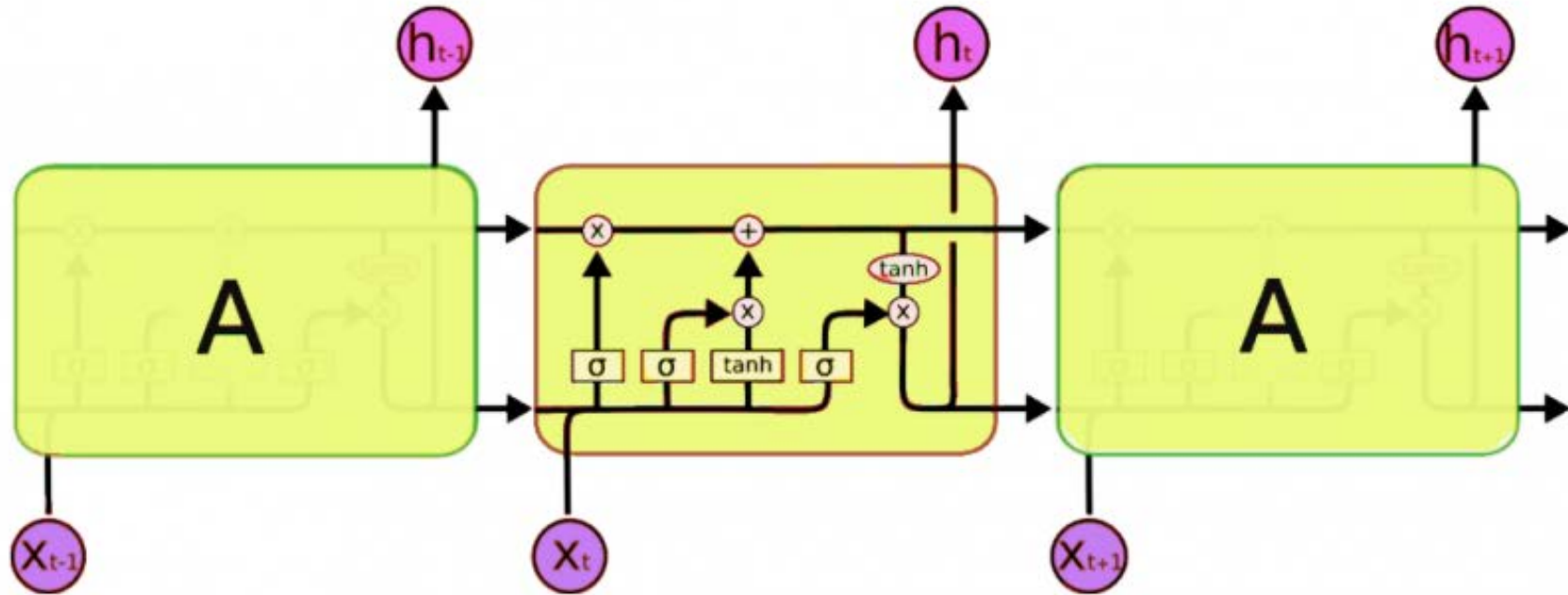
Model/Result

architecture
training process
result



Model-LSTM

- 為循環神經網路一種
- LSTM 是要改善 RNN 之缺點



Model-architecture

■ 將資料集轉化為監督學習問題

```
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
```


Model-architecture

■ 資料集劃分為訓練集和測試集

```
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,)

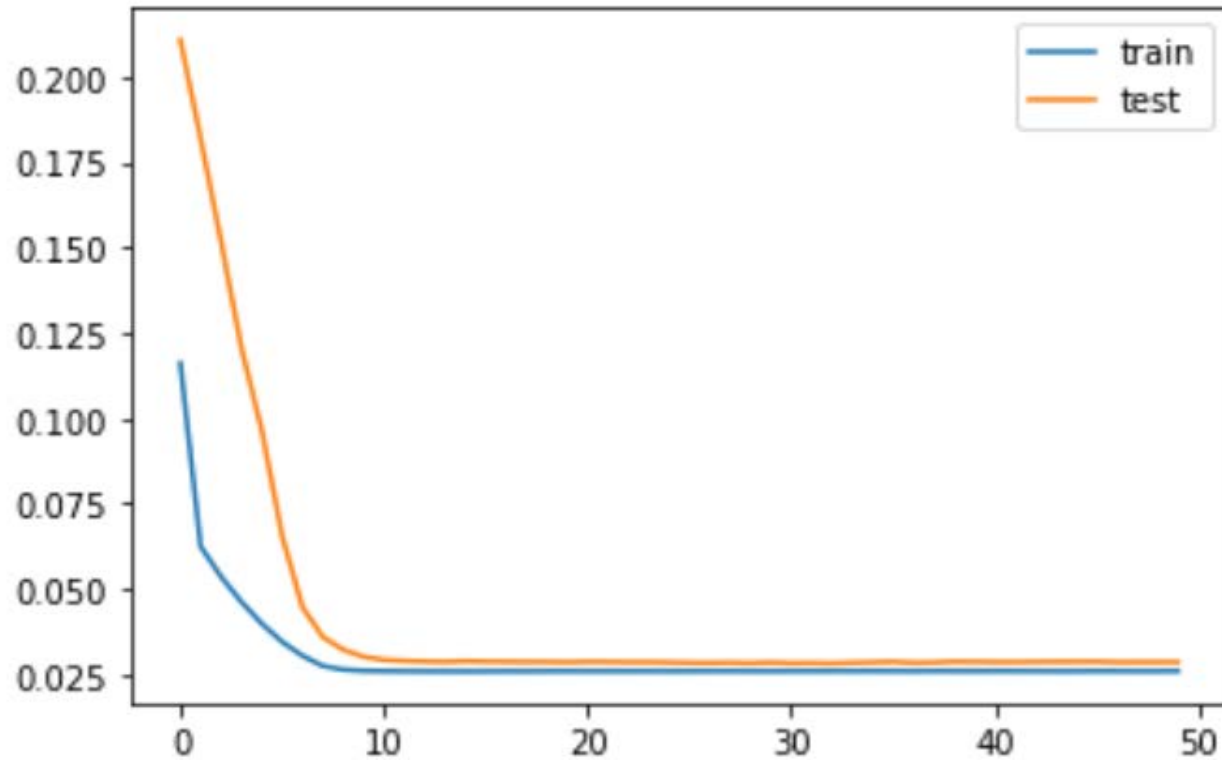
Model-architecture

■ 原始模型建立

```
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()
```

Model-architecture

■ 損失函數圖



Test RMSE: 6.513

Model-architecture

- 計算誤差百分比

```
import numpy  
  
me = numpy.mean( (inv_y - inv_yhat)/inv_yhat,axis = 0)  
  
print(abs(me))
```

0.10872321

Model-improve

■ 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()
```

RMSE 6.513

誤差率 0.10872321

Model-improve

■ 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()
```

RMSE 6.513 → 6.507

誤差率 0.10872321→0.11089778

Model-improve

■ 增加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()
```

RMSE 6.507→6.400

誤差率 0.11089778→0.07877497

Model-improve

■ 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()
```

RMSE 6.400→6.364

誤差率 0.07877497→0.076035604

Model-improve

■ 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()
```

RMSE 6.364→6.822

誤差率 0.076035604→0.14410850

Model-improve

■ 增加dropout

隱藏層內神經元：128

Epoch 數：50

優化演算法：adagrad

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()
```

RMSE 6.822→6.196

誤差率 0.14410850→0.042892274

Model-improve

■ 改善步驟

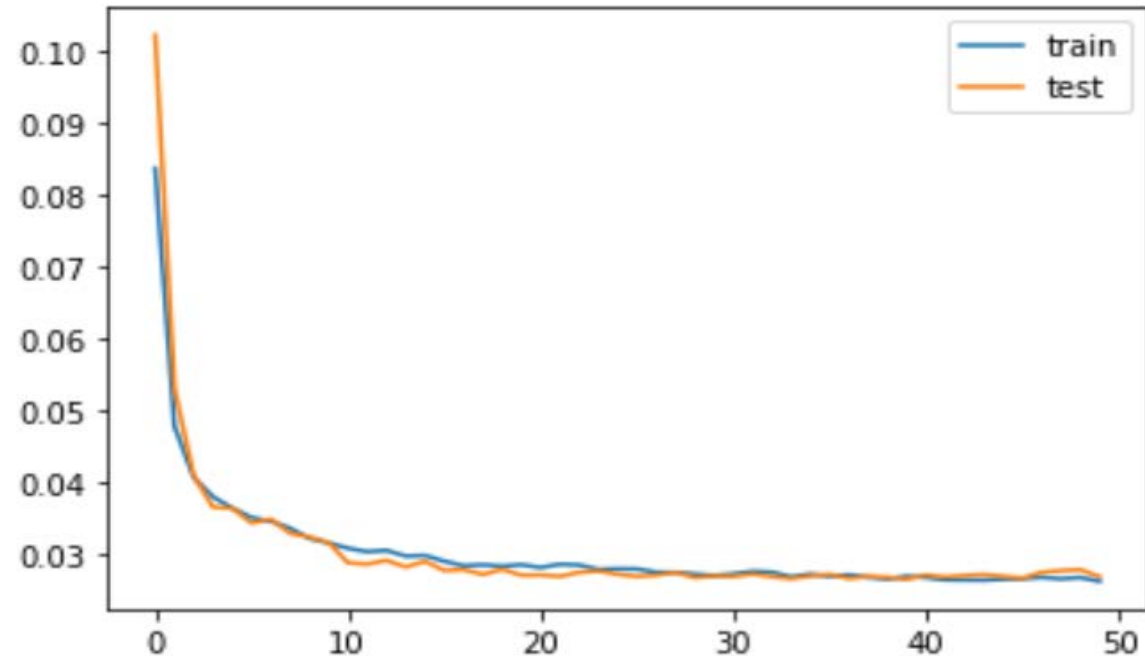
Step	神經元數	Epoch 數	優化演算法	丟棄率	誤差百分比
原	50	100	adam	-	10.87%
1	100	100	adam	-	11.09%
2	50	50	adam	-	7.88%
3	128	50	adam	-	7.6%
4	128	50	adagrad	-	14.41%
5	128	50	adam	0.3	4.29%

Result

■ 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
```

■ 最佳損失函數以及均方根誤差



Test RMSE: 6.196

■ 最低誤差百分比



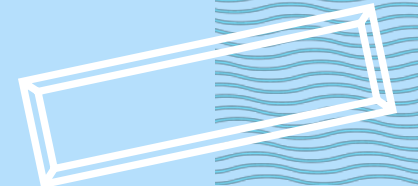
0.042892274



04

Conclusion

Discussion
Future work



Conclusion

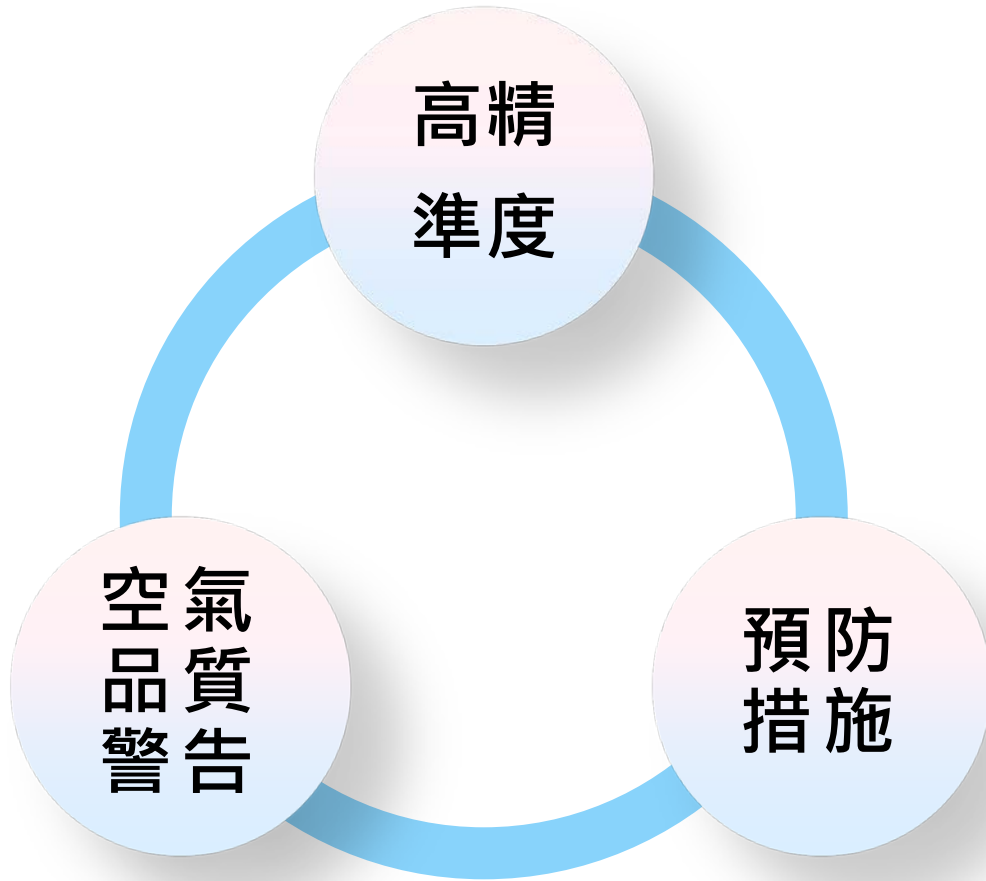
Discussion

資料修補方式
找尋更加適合的變量加入模型中訓練

Future work

結合 CNN 及 LSTM
數據視覺化
擴及全台(部分無觀測站區)

Conclusion



藉由LSTM建立具有較高預測能力的預測模型

預先對民眾提出空氣品質警告

提醒民眾事先做出預防措施，降低PM2.5對人體造成的傷害

Reference

Dataset : <https://www.epa.gov.tw/>

Model : 助教上課內容

<https://towardsdatascience.com/>

Content/Theme :

[1] 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.

[2] Saeed, S., Hussain, L., Awan, I. A. and Idris, A., 2017, Comparative analysis of different statistical methods for prediction of PM_{2.5} and PM₁₀ concentrations in advance for several hours, *International Journal of Computer Science and Network Security*, 17(11), 45-52.

...

T H A N K S

