機器學習期末報告

-第四組

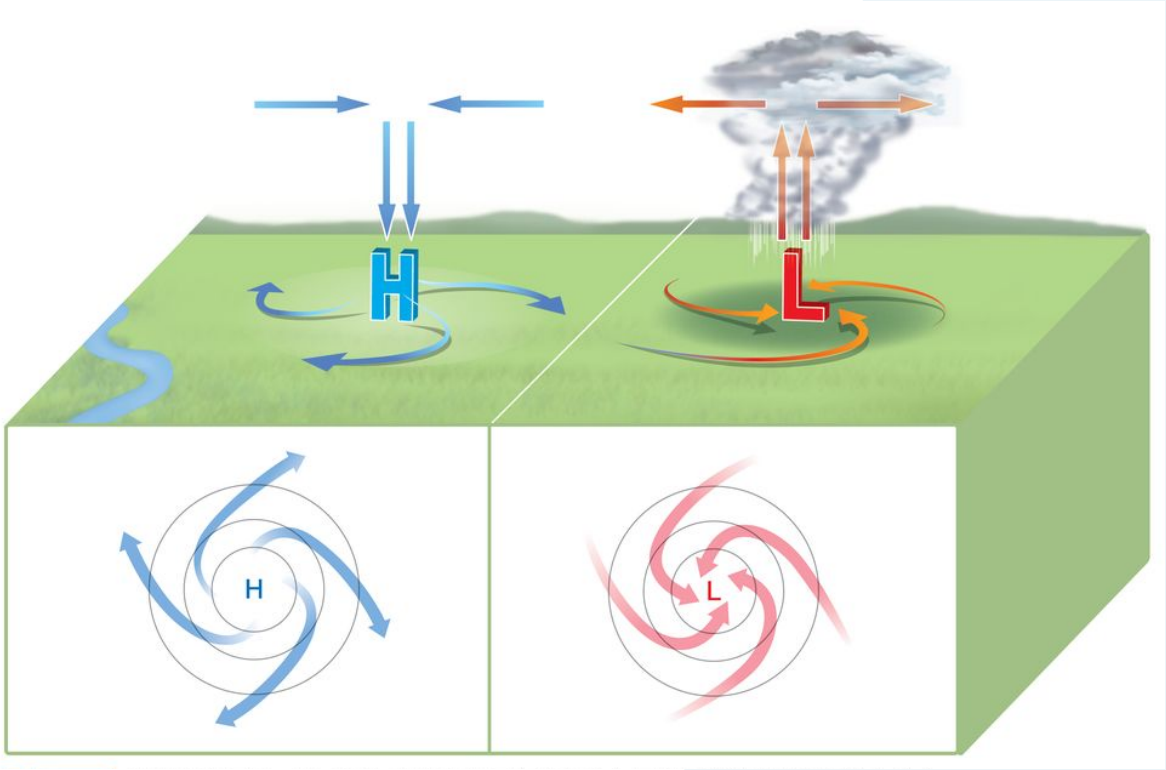
梅雨還是沒雨？

組員:109601001 王采琳 109601003 林群賀
109601501 黃于恩 109601508 林晴葳
109601510 吳彥叡

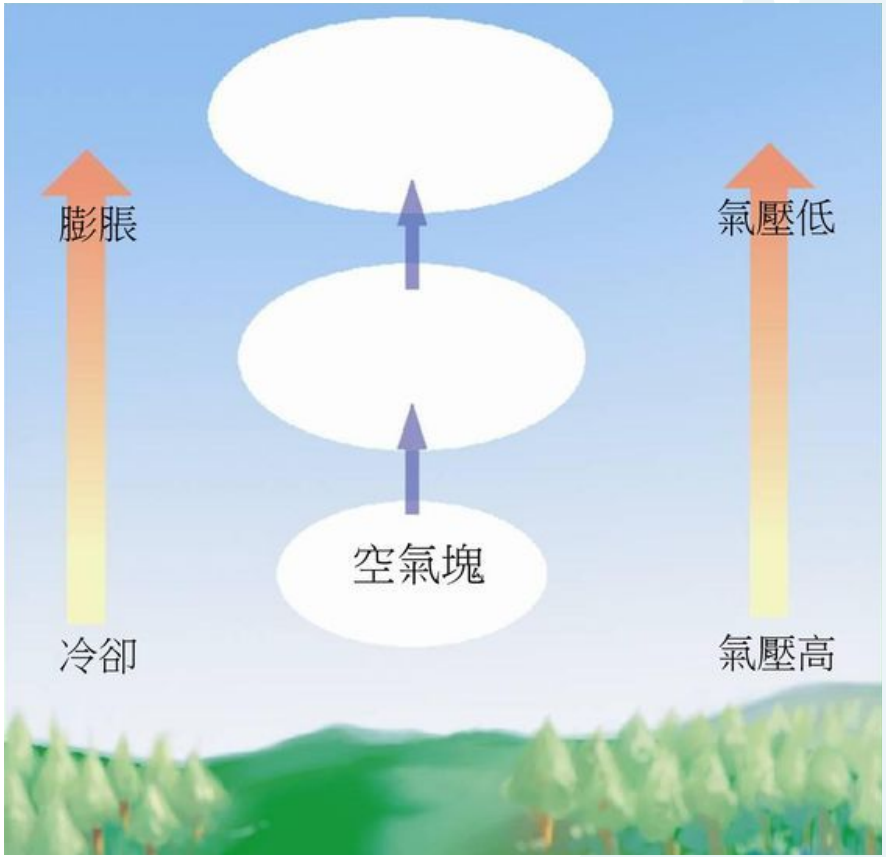
目錄

題目背景

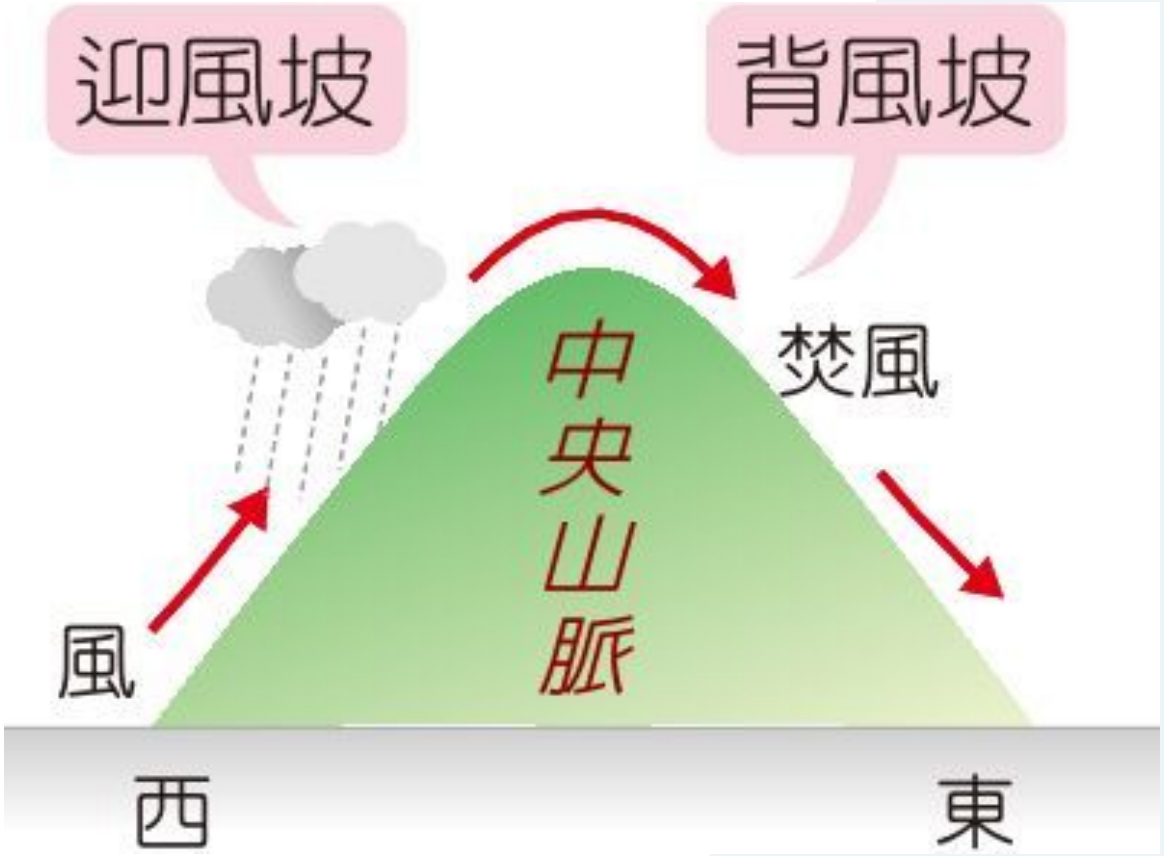
氣壓



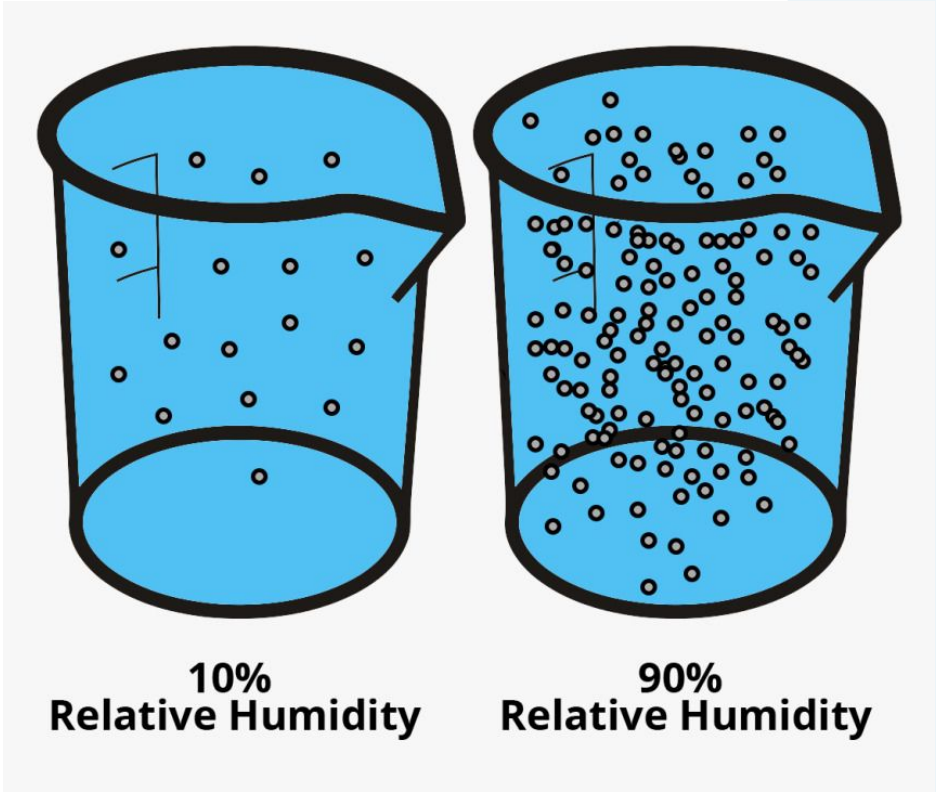
氣溫



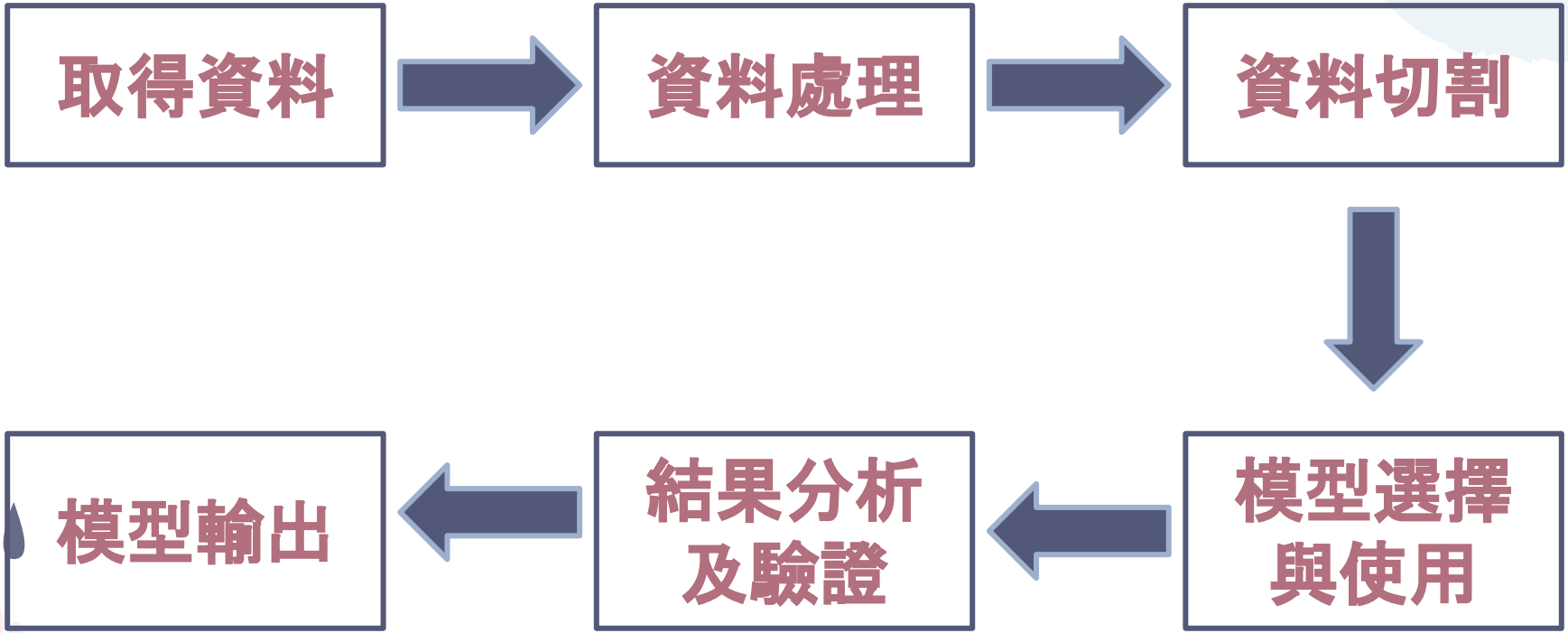
風



相對濕度

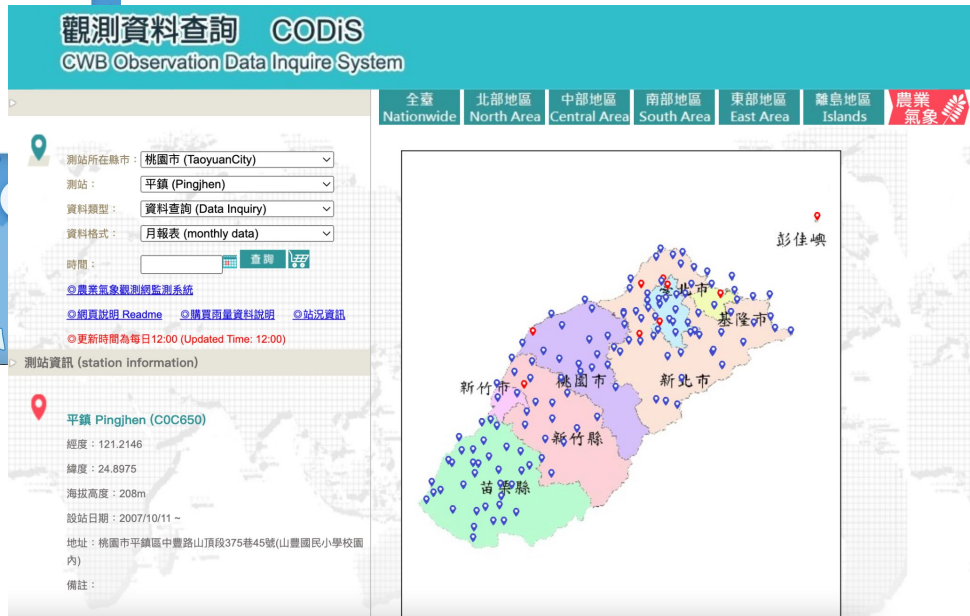


程式碼展示


取得資料
資料處理
資料切割
模型選擇與使用
結果分析及驗證
模型輸出

氣象局


交通部中央氣象局
Central Weather Bureau
回首頁 EN 網站導覽 意見箱 常見問答 關於本局 小 中 大
警特報 天氣 生活 地震 海象 氣候 資料 知識與天文 常用服務
資料
氣象產品目錄總集
氣象資料開放平臺
資料申購
每日雨量
每日雨水pH值
每日紫外線
觀測資料查詢
天文資料下載
研究出版與年報
桌布下載
氣象/海象預報模擬圖
政府資訊
政府資訊主動公開
氣象資源創新試用方案
檔案應用服務
內部控制制度聲明書
生態檢核專區
機關檔案目錄
全國法規資料庫
行政院公報
空氣品質監測 (測站名稱：忠明)
19
資料來源：環保署 (06/05 13:00:00)
縣市預報總覽
鄉鎮預報總覽
中央氣象局
數位科普網


觀測資料查詢 CODiS
CWB Observation Data Inquire System
全臺 Nationwide
北部地區 North Area
中部地區 Central Area
南部地區 South Area
東部地區 East Area
離島地區 Islands
農業氣象
測站所在縣市：桃園市 (TaoyuanCity)
測站：平鎮 (Pingjhen)
資料類型：資料查詢 (Data Inquiry)
資料格式：月報表 (monthly data)
時間：
查詢
◎農業氣象觀測網監測系統
◎網頁說明 Readme ◎購買雨量資料說明 ◎站況資訊
◎更新時間為每日12:00 (Updated Time: 12:00)
測站資訊 (station information)
平鎮 Pingjhen (C0C650)
經度：121.2146
緯度：24.8975
海拔高度：208m
設站日期：2007/10/11 ~
地址：桃園市平鎮區中豐路山頂段375巷45號(山豐國民小學校園內)
備註：
彭佳嶼
基隆市
新北市
新竹市
桃園市
新竹縣
苗栗縣

資料合併

```
import pandas as pd
import numpy as np
import os
import glob

path_domestic = os.path.abspath(os.getcwd()) + '/data'
data = glob.glob(os.path.join(path_domestic, "*.csv"))

df_1 = pd.concat((pd.read_csv(f) for f in data))

df_1.to_csv('data1/1_f.csv',encoding="utf_8_sig",index=False)
```

IMPORT 套件 / 讀取資料

```
import matplotlib.pyplot as plt
import seaborn as sns
```

利用已取得的資料，讀取並存入讀取並存入data frame

```
[2]  df = pd.read_csv("./1_f.csv")
```

觀察資料型態

```
[ ]  df.info()
     df.head()
     df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
ObsTime	854.0	15.754098	8.812117	1.0	8.0	16.0	23.0	31.0
Precp	854.0	8.038642	21.628585	0.0	0.0	0.0	4.0	312.0

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 854 entries, 0 to 853
Data columns (total 35 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ObsTime          854 non-null    int64
 1   StnPres          854 non-null    object
 2   SeaPres          854 non-null    object
 3   StnPresMax       854 non-null    object
 4   StnPresMaxTime   854 non-null    object
 5   StnPresMin       854 non-null    object
 6   StnPresMinTime   854 non-null    object
 7   Temperature      854 non-null    object
 8   T Max            854 non-null    object
 9   T Max Time       854 non-null    object
 10  T Min            854 non-null    object
 11  T Min Time       854 non-null    object
 12  Td dew point     854 non-null    object
 13  RH               854 non-null    object
 14  RHMin            854 non-null    object
 15  RHMinTime        854 non-null    object
 16  WS               854 non-null    object
 17  WD               854 non-null    object
 18  WSGust           854 non-null    object
 19  WDGust           854 non-null    object
 20  WGustTime        854 non-null    object
 21  Precp            854 non-null    float64
 26  PrecpMax60Time   854 non-null    object
 27  SunShine         854 non-null    object
 28  SunShineRate     854 non-null    object
 29  GloblRad         854 non-null    object
 30  VisbMean         854 non-null    object
 31  EvapA            854 non-null    object
 32  UVI Max          854 non-null    object
 33  UVI Max Time     854 non-null    object
 34  Cloud Amount     854 non-null    object
dtypes: float64(1), int64(1), object(33)
memory usage: 233.6+ KB
```

觀察資料

PrecpHour	PrecpMax10	*PrecpMax10Time* ↑	PrecpM…	PrecpM…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…

整理資料

```
df.drop(['ObsTime', 'SeaPres', 'StnPresMaxTime', 'StnPresMinTime'], axis = 1, inplace = True)
df.drop(['T Max Time', 'T Min Time', 'Td dew point'], axis = 1, inplace = True)
df.drop(['RHMinTime', 'WGustTime'], axis = 1, inplace = True)
df.drop(['PrecpHour', 'PrecpMax10', 'PrecpMax10Time', 'PrecpMax60', 'PrecpMax60Time'], axis = 1, inplace = True)
df.drop(['SunShine', 'SunShineRate', 'GloblRad', 'VisbMean'], axis = 1, inplace = True)
df.drop(['EvapA', 'UVI Max', 'UVI Max Time', 'Cloud Amount'], axis = 1, inplace = True)
```

再觀察一次資料

```
[ ]  df.info()
```

0 測站氣壓
1 測站氣壓最大值
2 測站氣壓最小值
3 氣溫
4 氣溫最大值
5 氣溫最小值
6 相對濕度
7 相對濕度最小值
8 風速
9 風向
10 陣風風速
11 陣風風向
12 降水量

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 854 entries, 0 to 853
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   StnPres      854 non-null    object
 1   StnPresMax   854 non-null    object
 2   StnPresMin   854 non-null    object
 3   Temperature  854 non-null    object
 4   T Max        854 non-null    object
 5   T Min        854 non-null    object
 6   RH           854 non-null    object
 7   RHMin        854 non-null    object
 8   WS           854 non-null    object
 9   WD           854 non-null    object
 10  WSGust       854 non-null    object
 11  WDGust       854 non-null    object
 12  Precp        854 non-null    float64
dtypes: float64(1), object(12)
memory usage: 86.9+ KB
```

轉換資料型態

```
df = pd.DataFrame(df, dtype = np.float)
```

```
cast.py:1181, in astype_nansafe(arr, dtype, copy, skipna)
   1177     raise ValueError(msg)
   1179 if copy or is_object_dtype(arr.dtype) or is_object_dtyp
e(dtype):
   1180     # Explicit copy, or required since NumPy can't view
from / to object.
-> 1181     return arr.astype(dtype, copy=True)
   1183 return arr.astype(dtype, copy=copy)

ValueError: could not convert string to float: '...'
```

轉換資料型態

Interactive-1.interactive > df (854, 13)

ıPre…	Temper…	T Max	T Min	RH	RHMin	WS	WD
	21.5	/	/	97	...	0.5	2
	20.4	/	/	95	...	1.5	8
7.1	20.0	23.1	17.7	91	76	0.7	11
7.1	21.3	24.6	19.5	94	79	0.3	283
5.7	22.2	26.2	20	94	69	0.3	225
	24.5	/	/	82	...	0.5	225
	25.6	/	/	75	...	0.4	33
2.8	26.1	32.4	21.7	68	41	0.6	17
1.4	27.3	32.3	21.3	61	43	0.6	351
9.5	28.1	33.7	23.8	62	39	2.9	216
	26.1	/	/	79	...	1.2	224
7.3	21.0	23.9	18.9	92	76	1.3	52
8.6	19.4	20.3	18.1	98	91	1.1	78
9.7	20.4	22	18.1	100	98	0.2	221
8.5	19.8	20.8	18.4	100	100	0.5	4
6.1	19.8	23.1	17.3	91	70	2.3	61
	23.7	/	/	74	...	0.6	8
	24.3	/	/	79	...	0.4	9
5.1	24.3	28.1	21.6	81	61	0.2	4
3.4	25.0	30.2	21.6	79	52	0.6	21

```
[ ]  df = df.replace('...','-999.0')
     df = df.replace('/','-999.0')
     df = pd.DataFrame(df, dtype = np.float)
```

轉換型態後再觀察一次

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 854 entries, 0 to 853
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   StnPres      854 non-null    float64
 1   StnPresMax   854 non-null    float64
 2   StnPresMin   854 non-null    float64
 3   Temperature  854 non-null    float64
 4   T Max        854 non-null    float64
 5   T Min        854 non-null    float64
 6   RH           854 non-null    float64
 7   RHMin        854 non-null    float64
 8   WS           854 non-null    float64
 9   WD           854 non-null    float64
 10  WSGust       854 non-null    float64
 11  WDGust       854 non-null    float64
 12  Precp        854 non-null    float64
dtypes: float64(13)
memory usage: 86.9 KB
```

處理降雨量數值

```
for k in range(854):
    if df.iloc[k,12] > 0.0:
        df.iloc[k,12] = 1
    else:
        df.iloc[k,12] = 0
```

Precp
0.5
0.5
0
0
60.5
19.5
19
0
0
4
0
0
0

Precp
1
1
0
0
1
1
1
0
0
1
0
0
0

填補缺失值　取平均

```
for k in range(0,853):
    if (df.iloc[k,0] != -999.0):
        stnpres = float(df.iloc[k,0])
        count0 += 1
        stnprestotal += stnpres
    if (df.iloc[k,1] != -999.0):
        stnpresmax = float(df.iloc[k,1])
        count1 += 1
        stnpresmaxtotal += stnpresmax
    if (df.iloc[k,2] != -999.0):
        stnpresmin = float(df.iloc[k,2])
        count2 += 1
        stnpresmintotal += stnpresmin
    if (df.iloc[k,3] != -999.0):
        T = float(df.iloc[k,3])
        count3 += 1
        Ttotal += T
```

```
    if (df.iloc[k,4] != -999.0):
        Tmax = float(df.iloc[k,4])
        count4 += 1
        Tmaxtotal += Tmax
    if (df.iloc[k,5] != -999.0):
        Tmin = float(df.iloc[k,5])
        count5 += 1
        Tmintotal += Tmin
    if (df.iloc[k,8] != -999.0):
        WS = float(df.iloc[k,8])
        count8 += 1
        WStotal += WS
    if (df.iloc[k,10] != -999.0):
        WSGust = float(df.iloc[k,10])
        count10 += 1
        WSGusttotal += WSGust
```

氣壓 StnPres	氣壓最大值 StnPresMax	氣壓最小值 StnPreMin	氣溫 Temperature	氣溫最大值 T Max	氣溫最小值 T Min	陣風風速 WSGust	風速 WS
平均數							

填補缺失值　取平均

```
ave0  =  round(stnprestotal  /  count0  ,  1)
ave1  =  round(stnpresmaxtotal  /  count1  ,  1)
ave2  =  round(stnpresmintotal  /  count2  ,  1)
ave3  =  round(Ttotal  /  count3  ,  1)
ave4  =  round(Tmaxtotal  /  count4  ,  1)
ave5  =  round(Tmintotal  /  count5  ,  1)
ave8  =  round(WStotal  /  count8  ,  1)
ave10  =  round(WSGusttotal  /  count10  ,  1)
```

```
for  c  in  range(854):
        if  df.iloc[c,0]  ==  -999.0:
            df.iloc[c,0]  =  ave0
        if  df.iloc[c,1]  ==  -999.0:
            df.iloc[c,1]  =  ave1
        if  df.iloc[c,2]  ==  -999.0:
            df.iloc[c,2]  =  ave2
        if  df.iloc[c,3]  ==  -999.0:
            df.iloc[c,3]  =  ave3
        if  df.iloc[c,4]  ==  -999.0:
            df.iloc[c,4]  =  ave4
        if  df.iloc[c,5]  ==  -999.0:
            df.iloc[c,5]  =  ave5
        if  df.iloc[c,8]  ==  -999.0:
            df.iloc[c,8]  =  ave8
        if  df.iloc[c,10]  ==  -999.0:
            df.iloc[c,10]  =  ave10
```

填補缺失值　取眾數

相對濕度 RH	相對溼度最小值 RHMin	風向 WD	陣風風向 WDGust
眾數			

```
for i in range(854):
    if df.iloc[i,6] == -999.0:
        df.iloc[i,6] = df['RH'].value_counts().idxmax()

for i in range(854):
    if df.iloc[i,7] == -999.0:
        df.iloc[i,7] = df['RHMin'].value_counts().idxmax()

for i in range(854):
    if df.iloc[i,9] == -999.0:
        df.iloc[i,9] = df['WD'].value_counts().idxmax()

for i in range(854):
    if df.iloc[i,11] == -999.0:
        df.iloc[i,11] = df['WDGust'].value_counts().idxmax()
```

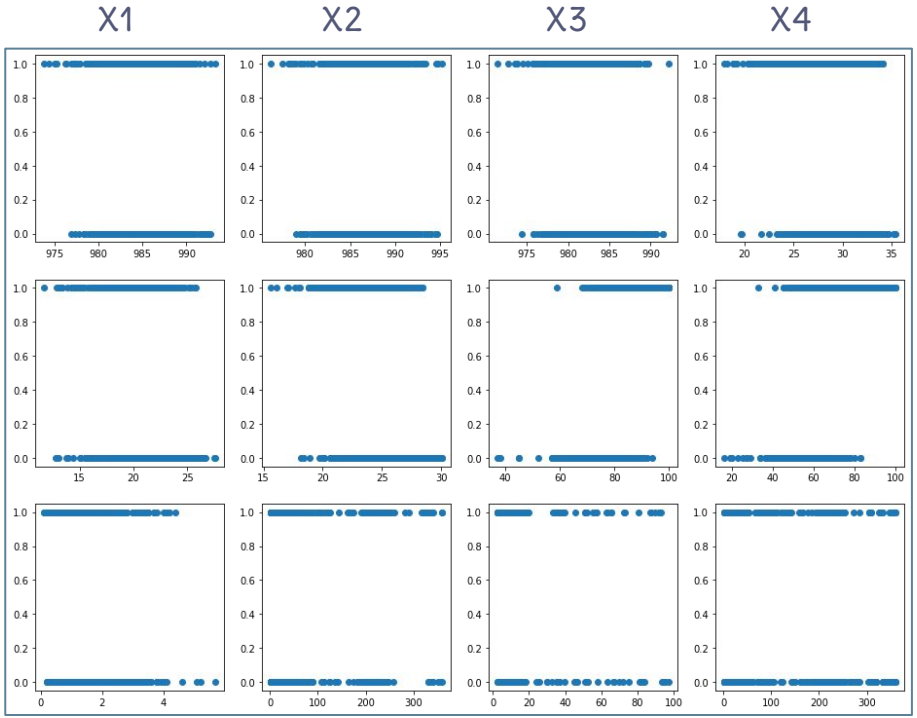
1. 我們認為眾數較能代表大部分時間的溼度，如果取平均的話比較容易出現不符合實際狀況的數 值
2. 風向是以方位角表達，所以當然不能取平均，眾數較適合

作圖觀察各個變數和降雨量的關係

1. 氣壓很低的時候通常會下雨
2. 溫度低的時候通常是下雨天
3. 相對濕度很高的時候通常會下雨
4. 風速很高的時候通常不會下雨

```
x1  =  df['StnPres']
x2  =  df['StnPresMax']
x3  =  df['StnPresMin']
x4  =  df['T  Max']
x5  =  df['T  Min']
x6  =  df['Temperature']
x7  =  df['RH']
```

```
x8  =  df['RHMin']
x9  =  df['WS']
x10  =  df['WD']
x11  =  df['WSGust']
x12  =  df['WDGust']
y  =  df['Precp']
```



觀察各變數之間的相關係數

```
[ ]  df.corr()
```

➔ 相對溼度、氣溫和降水量的相關性最強烈

	StnPres	StnPresMax	StnPresMin	Temperature	T Max	T Min	RH	RHMin	WS	WD	WSGust	WDGust	Precp
StnPres	1.000000	0.973837	0.977780	-0.393330	-0.278551	-0.365425	-0.000452	0.002457	-0.095994	-0.205172	-0.129541	-0.191010	-0.102302
StnPresMax	0.973837	1.000000	0.942063	-0.401371	-0.286646	-0.389644	0.006529	0.006868	-0.063372	-0.182046	-0.106577	-0.185542	-0.068398
StnPresMin	0.977780	0.942063	1.000000	-0.360691	-0.258129	-0.332116	-0.012693	-0.009457	-0.107386	-0.194920	-0.130746	-0.188141	-0.114482
Temperature	-0.393330	-0.401371	-0.360691	1.000000	0.936330	0.800535	-0.429906	-0.462815	-0.133649	0.577173	0.086536	0.420318	-0.436195
T Max	-0.278551	-0.286646	-0.258129	0.936330	1.000000	0.691656	-0.521878	-0.640560	-0.128305	0.530669	0.103263	0.386274	-0.465951
T Min	-0.365425	-0.389644	-0.332116	0.800535	0.691656	1.000000	-0.152051	-0.159250	-0.102209	0.400518	0.094228	0.329499	-0.223502
RH	-0.000452	0.006529	-0.012693	-0.429906	-0.521878	-0.152051	1.000000	0.882385	-0.341775	-0.288269	-0.141324	-0.218402	0.635275
RHMin	0.002457	0.006868	-0.009457	-0.462815	-0.640560	-0.159250	0.882385	1.000000	-0.189154	-0.305214	-0.184937	-0.234054	0.534163
WS	-0.095994	-0.063372	-0.107386	-0.133649	-0.128305	-0.102209	-0.341775	-0.189154	1.000000	-0.053436	0.044637	-0.091865	-0.094960
WD	-0.205172	-0.182046	-0.194920	0.577173	0.530669	0.400518	-0.288269	-0.305214	-0.053436	1.000000	0.082668	0.422772	-0.255363
WSGust	-0.129541	-0.106577	-0.130746	0.086536	0.103263	0.094228	-0.141324	-0.184937	0.044637	0.082668	1.000000	0.114226	0.022863
WDGust	-0.191010	-0.185542	-0.188141	0.420318	0.386274	0.329499	-0.218402	-0.234054	-0.091865	0.422772	0.114226	1.000000	-0.212534
Precp	-0.102302	-0.068398	-0.114482	-0.436195	-0.465951	-0.223502	0.635275	0.534163	-0.094960	-0.255363	0.022863	-0.212534	1.000000

資料切割、模型選擇

我們接著再把所有資料都餵給模型，來觀察模型辨識程度

訓練:70%
測試:30%

```
X = df.drop(['Precp'], axis=1)
y = df['Precp']

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score

X_train,X_test, y_train,y_test = train_test_split(X, y, test_size=0.3, random_state=67)
lr = LogisticRegression(max_iter=200)
lr.fit(X_train, y_train)
predictions = lr.predict(X_test)
accuracy_score(y_test, predictions)
recall_score(y_test, predictions)
precision_score(y_test, predictions)
```

資料切割改善-測試1

```
X= df.drop(['Precp'],axis=1)
y= df['Precp']
```

✓ accuracy_score(y_test, predictions) ···

```
0.8287937743190662
```

✓ recall_score(y_test, predictions) ···

```
0.6966292134831461
```

✓ precision_score(y_test, predictions) ···

```
0.7848101265822784
```

✓ pd.DataFrame(confusion_matrix(y_test, ···

	Predict not Precp	Predict Precp
True not Precp	151	17
True Precp	27	62

資料切割改善-測試2

```
X= df.drop(['Precp','StnPresMax','StnPresMin','T Max','T Min','RHMin'],axis=1)
y= df['Precp']
```

#出來的結果沒有比1好

✓ accuracy_score(y_test, predictions) ···

```
0.7976653696498055
```

✓ recall_score(y_test, predictions) ···

```
0.6629213483146067
```

✓ precision_score(y_test, predictions) ···

```
0.7283950617283951
```

✓ pd.DataFrame(confusion_matrix(y_test, ···

	Predict not Precp	Predict Precp
True not Precp	146	22
True Precp	30	59

資料切割改善-測試3

```
X= df.drop(['Precp','WD','WDGust'],axis=1)
y= df['Precp']
```

#比2好

```
✓ accuracy_score(y_test, predictions) ···
```

```
0.8171206225680934
```

```
✓ recall_score(y_test, predictions) ···
```

```
0.6853932584269663
```

```
✓ precision_score(y_test, predictions) ···
```

```
0.7625
```

```
✓ pd.DataFrame(confusion_matrix(y_test,
```

	Predict not Precp	Predict Precp
True not Precp	149	19
True Precp	28	61

資料切割改善-測試4

```
X= df.drop(['Precp','WD','WS','WDGust','WSGust'],axis=1)
y= df['Precp']
```

#結果比3差

✓ accuracy_score(y_test, predictions) ···

```
0.8054474708171206
```

✓ recall_score(y_test, predictions) ···

```
0.6853932584269663
```

✓ precision_score(y_test, predictions) ···

```
0.7349397590361446
```

✓ pd.DataFrame(confusion_matrix(y_test, ···

	Predict not Precp	Predict Precp
True not Precp	146	22
True Precp	28	61

資料切割改善-測試5

```
X= df.drop(['Precp','T Max','T Min'],axis=1)
y= df['Precp']
```

#結果比3、4差，比2好

```
✓ accuracy_score(y_test, predictions) ···
```

```
0.8015564202334631
```

```
✓ recall_score(y_test, predictions) ···
```

```
0.6741573033707865
```

```
✓ precision_score(y_test, predictions) ···
```

```
0.7317073170731707
```

```
✓ pd.DataFrame(confusion_matrix(y_test,  ···
```

	Predict not Precp	Predict Precp
True not Precp	146	22
True Precp	29	60

資料切割改善-測試6

```
X= df.drop(['Precp','WD','WDGust','T Max','T Min'],axis=1)
y= df['Precp']
```

#結果跟5差不多

✓ accuracy_score(y_test, predictions) ···

```
0.8015564202334631
```

✓ recall_score(y_test, predictions) ···

```
0.6629213483146067
```

✓ precision_score(y_test, predictions) ···

```
0.7375
```

✓ pd.DataFrame(confusion_matrix(y_test, ···

	Predict not Precp	Predict Precp
True not Precp	147	21
True Precp	30	59

資料切割改善-測試7

```
X= df.drop(['Precp','WSGust','WDGust','T Max','T Min'],axis=1)
y= df['Precp']
```

#結果跟6一樣

✓ accuracy_score(y_test, predictions) ···

0.8015564202334631

✓ recall_score(y_test, predictions) ···

0.6629213483146067

✓ precision_score(y_test, predictions) ···

0.7375

✓ pd.DataFrame(confusion_matrix(y_test, ···

	Predict not Precp	Predict Precp
True not Precp	147	21
True Precp	30	59

結果分析與驗證

經過多次交叉測試後，發現什麼都沒有刪除，分數會最高

```
X = df.drop(['Precp'], axis=1)
y = df['Precp']
X_train,X_test, y_train,y_test = train_test_split(X, y, test_size=0.3, random_state=67)
lr = LogisticRegression(max_iter=200)
lr.fit(X_train, y_train)
predictions = lr.predict(X_test)
accuracy_score(y_test, predictions)
recall_score(y_test, predictions)
precision_score(y_test, predictions)
```

結果分析與驗證

最後我們測試模型的辨認是否順利，然後有發現到測出來的型態為[1.]、[0.]

```
print(lr.predict([[900,  1000,  850,  23,  27,  18,  34,  12,  1,  23,  2,  45]]))
print(lr.predict([[900,  860,  950 ,  26,  31,  20,  70 ,  50 ,  3 ,  20 ,  6 ,  25 ]]))
```

模型輸出

利用joblib模組將模型匯出

```
import  joblib
joblib.dump(lr,'Precipitation_Predict.pkl',compress=3)
```

WEB

code

```
6   import joblib
7   import numpy as np
8   from flask import Flask, render_template, request, url_for
9   import os
10
11  modelPretrained = joblib.load("./Precipitation_Predict.pkl")
12  app = Flask(__name__)
13
14  @app.route('/')
15  def formPage():
16      return render_template("form.html")
```

code

```python
@app.route("/submit", methods = ['POST'])
def submit():

    if request.method == 'POST':
        form_data = request.form
        # print(form_data)


        inputPressure = ""
        relativeHumidity = ""
        wind = ""
        gustWind = ""
        inputTemerature = ""
```

code

```
34          result = modelPretrained.predict([[
35          int(form_data["currentPressure"]),
36          int(form_data["todayMaximumPressure"]),
37          int(form_data["todayMinimumPressure"]),
38          int(form_data["currentTemperature"]),
39          int(form_data["todayMaximumTemperature"]),
40          int(form_data["todayMinimumTemperature"]),
41          int(form_data["currentRelativeHumidity"]),
42          int(form_data["todayMinimumRelativeHumidity"]),
43          int(form_data["currentWindSpeed"]),
44          int(form_data["currentWindDirection"]),
45          int(form_data["currentGustWindSpeed"]),
46          int(form_data["currentGustWindDirection"])
47          ]])
```

code

```
49          resultProba = modelPretrained.predict_proba([[
50          int(form_data["currentPressure"]),
51          int(form_data["todayMaximumPressure"]),
52          int(form_data["todayMinimumPressure"]),
53          int(form_data["currentTemperature"]),
54          int(form_data["todayMaximumTemperature"]),
55          int(form_data["todayMinimumTemperature"]),
56          int(form_data["currentRelativeHumidity"]),
57          int(form_data["todayMinimumRelativeHumidity"]),
58          int(form_data["currentWindSpeed"]),
59          int(form_data["currentWindDirection"]),
60          int(form_data["currentGustWindSpeed"]),
61          int(form_data["currentGustWindDirection"])
62          ]])
```

code

```
64      print(f'Result:{result}')
65      print(f'Result_Proba:{resultProba}')
66
67      if result[0] == 1.:
68          prediction = f'會下雨哦！ - 系統信心 {resultProba[0][1]:.10f}'
69      else:
70          prediction = f'不會下雨哦！ - 系統信心 {resultProba[0][0]:.10f}'
```

code

```
73          return render_template('form.html',
74          inputPressure = inputPressure,
75          currentPressure = form_data["currentPressure"],
76          todayMaximumPressure = form_data["todayMaximumPressure"],
77          todayMinimumPressure = form_data["todayMinimumPressure"],
78          inputTemerature = inputTemerature,
79          currentTemperature = form_data["currentTemperature"],
80          todayMaximumTemperature = form_data["todayMaximumTemperature"],
81          todayMinimumTemperature = form_data["todayMinimumTemperature"],
82          relativeHumidity = relativeHumidity,
83          currentRelativeHumidity = form_data["currentRelativeHumidity"],
84          todayMinimumRelativeHumidity = form_data["todayMinimumRelativeHumidity"],
85          wind = wind,
86          currentWindSpeed = form_data["currentWindDirection"],
87          currentWindDirection = form_data["currentWindDirection"],
88          gustWind = gustWind,
89          currentGustWindSpeed = form_data["currentGustWindSpeed"],
90          currentGustWindDirection = form_data["currentGustWindDirection"],
91          prediction = prediction)
```

code

```
86  if __name__ == "__main__":
87      app.run()
```

WEB Demo

結論

改善

心得

Reference

1. https://blog.csdn.net/wushaowu2014/article/details/78963709
2. https://www.csie.ntu.edu.tw/~r91112/myDownload/WEB/html.html
3. https://medium.com/seaniap/python-web-flask-如何透過form取得資料-7a63ebf9ff1f
4. https://stackoverflow.com/questions/48780324/flask-bad-request-the-browser-or-proxy-sent-a-request-that-this-server-could
5. https://stackoverflow.com/questions/42126772/typeerror-init-got-an-unexpected-keyword-argument-method
6. https://www.uj5u.com/ruanti/262589.html
7. https://www.cwb.gov.tw/V8/C/
8. https://blog.csdn.net/wushaowu2014/article/details/78963709

Appendix

1. Source Code：https://colab.research.google.com/drive/1-EH2SZtS_Zc55DUc7PGL7xN58DahNL94?hl=zh-tw#scrollTo=4ZRYCWMPLz4d

DIVIDE AND CONQUER

林群賀

網頁、簡報

王采琳

資料前處理、程式碼、簡報

林晴葳

資料前處理、程式碼、簡報

黃于恩

資料前處理、程式碼、簡報

吳彥叡

程式碼、簡報

THANKS!