登山攻頂成功機率預測模型之建構與驗證

Group 5 107034703 陳家安
108034537 黃子芸
108034540 吳欣蓉
108034545 黃唯真

指導教授：邱銘傳 教授

Agenda

01 研究主題

5W1H

02 資料分析與前處理

資料來源與結構分析與資料前處理

03 模型架構

模型架構、應用方法、參數設定與模型訓練

04 模型效度驗證

05 小結與未來展望

主題說明_登山服務企業

What 蒐集天氣資料及登山成功率等歷史資料，
預測未來登山的成功機率

Where Mount Rainier，分析山間天氣與各登山路線資料

Who 登山客、登山數據預測團隊

When 登山客預計登山日期

Why 透過分析登山路線與天氣等影響因素，提供登山客欲攻頂日的成功機率，
為登山客考量是否應該改期登山，降低登山風險

How 登山路線與天氣狀況的資料前處理及DNN模型訓練

資料來源與結構分析

- 資料來源：Nation Park Service官方網站、公開資料集
- 取得Mount Rainier(瑞尼爾山)共21條登山路線：
  登頂嘗試次數和成功率的歷史資料

Date	Battery Vo	Temperature AVG	Relative Humidity AVG	Wind Speed Daily AVG	Wind Direction AVG	Solare Radi
12/31/2015	13.845	19.06291667	21.87083333	21.97779167	62.32583333	84.91529
12/30/2015	13.82292	14.63120833	18.49383333	3.540541667	121.5054167	86.19283
12/29/2015	13.83458	6.614291667	34.07291667	0	130.2916667	85.10092
12/28/2015	13.71042	8.687041667	70.55791667	0	164.68375	86.24125
12/27/2015	13.3625	14.14041667	95.75416667	0	268.4791667	31.09071

- 山間的天氣歷史資料：
  包含平均氣溫、平均濕度、平均風速、平均太陽輻射等

Date	Battery Vo	Temperature AVG	Relative Humidity AVG	Wind Speed Daily AVG	Wind Direction AVG	Solare Radi
12/31/2015	13.845	19.06291667	21.87083333	21.97779167	62.32583333	84.91529
12/30/2015	13.82292	14.63120833	18.49383333	3.540541667	121.5054167	86.19283
12/29/2015	13.83458	6.614291667	34.07291667	0	130.2916667	85.10092
12/28/2015	13.71042	8.687041667	70.55791667	0	164.68375	86.24125
12/27/2015	13.3625	14.14041667	95.75416667	0	268.4791667	31.09071

資料前處理

Step1：先將兩份資料檔(登山路線與天氣狀況)依據日期合併，列出合併結果

Code >

```
# Merge climbing and weather datasets on the Date columns
dfm = df.merge(wth, on='Date')
# Then, sort Date values ascending and reset index
dfm.sort_values('Date', inplace=True)
dfm.reset_index(drop=True, inplace=True)
dfm.head(3)
dfm.info()
dfm.describe(include='all')
```

Output >

	Date	Route	Attempted	Succeeded	Success Percentage	Battery Voltage AVG	Temperature AVG	Relative Humidity AVG
0	2014-09-23	Disappointment Cleaver	11	0	0.0	13.056667	32.857333	100.000000
1	2014-09-24	Disappointment Cleaver	12	0	0.0	13.168750	29.702917	100.000000
2	2014-09-25	Disappointment Cleaver	2	2	1.0	13.648333	26.823750	99.854167

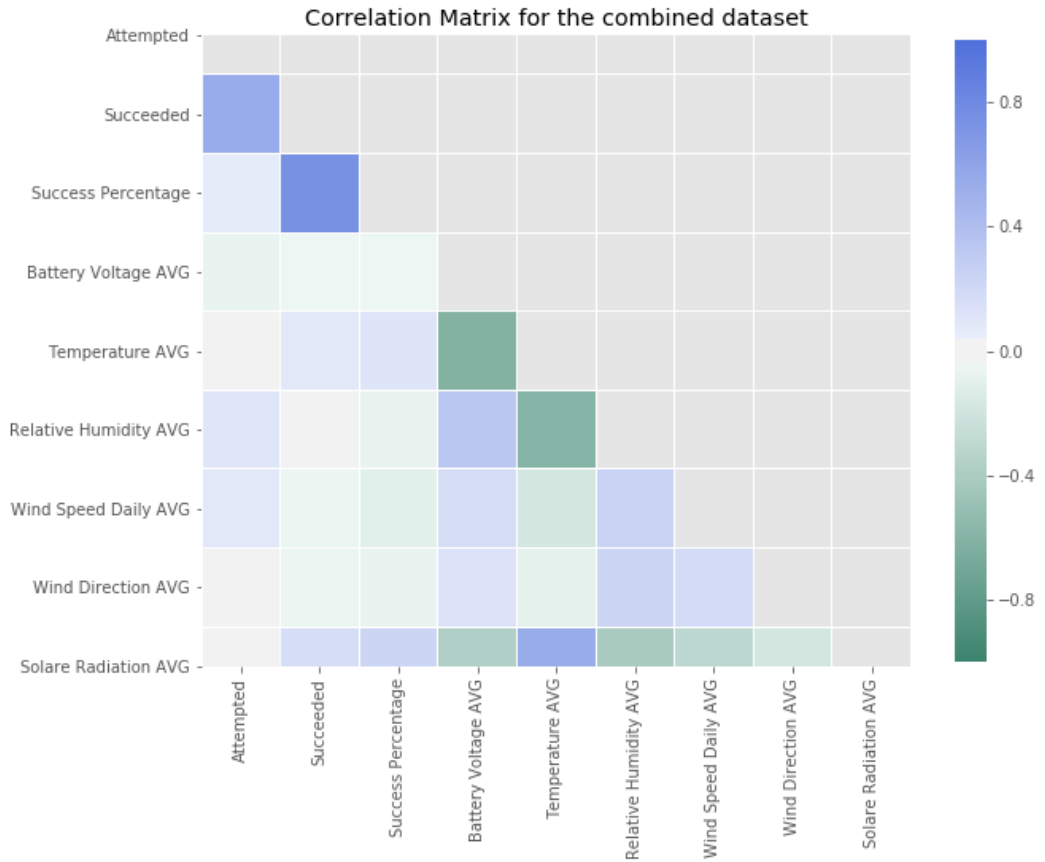
資料前處理

Step2：將資料中的參數進行參數間相關性的比較，做出相關矩陣圖

Code >

```
[12] # Now plot a correlation matrix:

     corr = df_countable.corr()
     mask = np.zeros_like(corr, dtype=np.bool)
     mask[np.triu_indices_from(mask)] = True

     f, ax = plt.subplots(figsize=(10, 9))

     cmap = sns.diverging_palette(210, 5, as_cmap=True)

     sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0,
                 square=True, linewidths=.1, cbar_kws={"shrink": .8})
     plt.title('Correlation Matrix for the combined dataset')

     plt.show()
```

Output >



資料前處理

Step3：數據修正與調整

1.

由原始資料中我們發現，「成功機率(Success Percentage)」有14.2的值，這顯然是一個異常值（機率應介於0到1之間），按以下規則進行數據修改：數值> 1時，將其設為1。

2.

將成功次數大於嘗試次數的數據，改為成功次數 = 嘗試次數

「成功次數(Succeeded)」和「成功機率(Success Percentage)」之間（在邏輯上）有很強的相關性，因此我們將刪除「成功次數」那欄的數據。

3.

「Battery Voltage AVG」表示電池電壓，並且似乎與平均溫度負相關， 但我們選擇保留此列。

4.

資料前處理

Step3：數據修正與調整

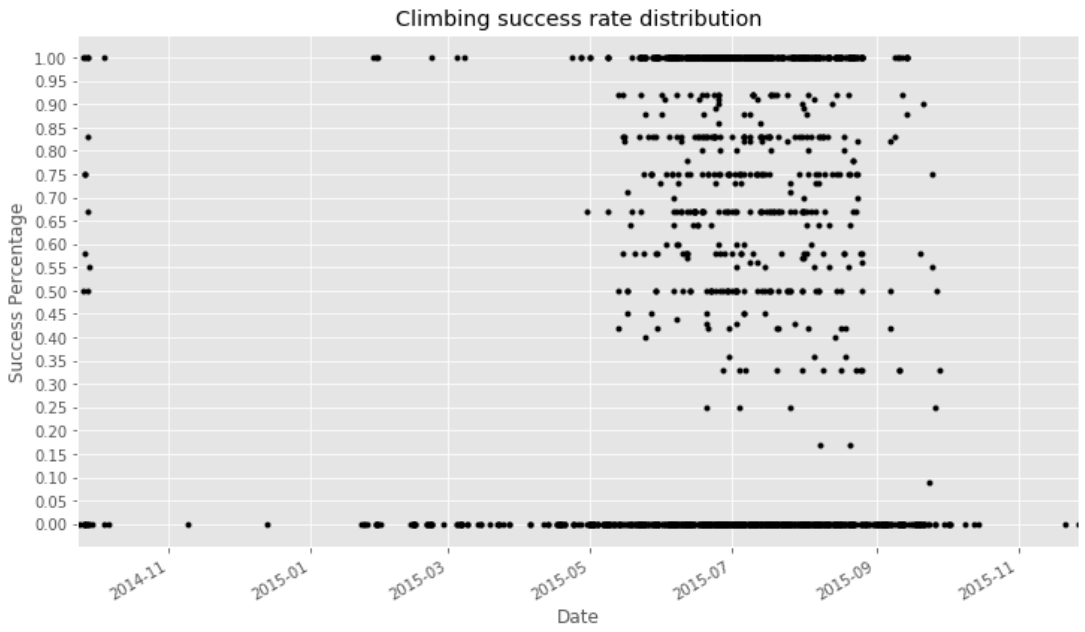
Code >

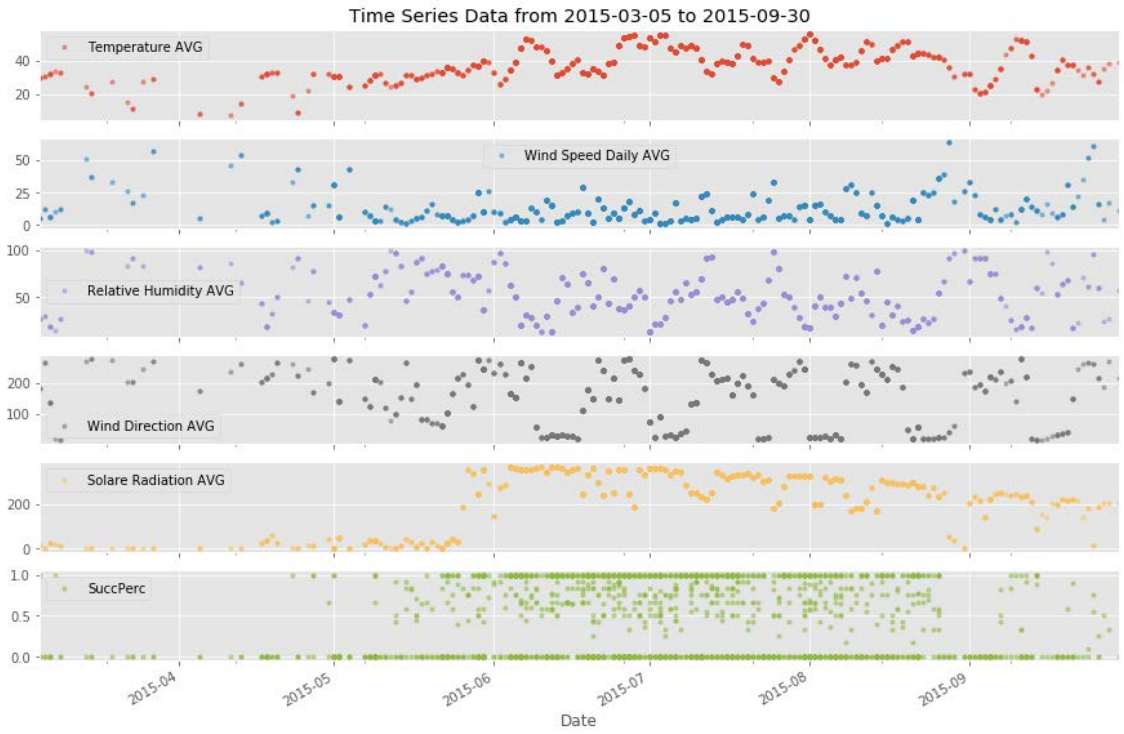
```
# Let's first rename the target feature column (for convenience purpose only):
dfm.rename(columns={'Success Percentage' : 'SuccPerc'}, inplace=True)

# Bring 'Succeeded' to be equal 'Attempted' where 'Succeeded' > 'Attempted'
dfm.loc[dfm['Succeeded'] > dfm['Attempted'], 'Succeeded'] = dfm['Attempted']

# Now, locate the outliers and bring them to 1.
dfm.loc[dfm['SuccPerc'] > 1, 'SuccPerc'] = 1
dfm['SuccPerc'] = dfm['SuccPerc'].round(2)
```

資料前處理

Step4：將數據的日期設為索引，畫出日期與成功機率以及其他因子的分佈圖

Output >





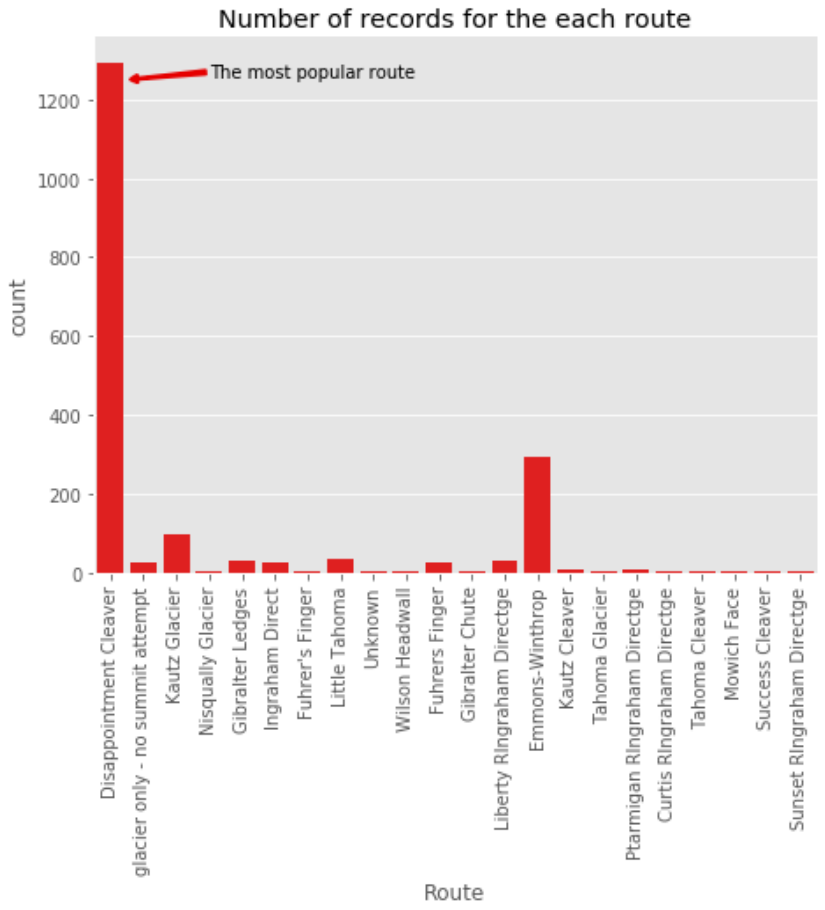
資料前處理

Step5：統計各登山路線的登山次數，將重複的路線資訊合併

Code >

```
g = sns.catplot(x = 'Route', data=dfm, aspect=1.5, kind="count", color="r")
g.set_xticklabels(rotation=90)
plt.title('Number of records for the each route')
plt.annotate('The most popular route',
             xy=(3,1250),
             rotation=0,
             va='bottom',
             ha='left',
            )

plt.annotate('',
             xy=(0.5, 1250),
             xytext=(3, 1270),
             xycoords='data',
             arrowprops=dict(arrowstyle='fancy ,head_length=0.4,head_width=0.4,tail_width=0.2',
                             connectionstyle='arc3',
                             color='xkcd:red',
                             lw=2
                            ))
plt.show()
```

< Output



模型架構

建構步驟與參數調整

建構步驟與方法

資料前處理 → 模型建立 → 參數調整 → 模型訓練

每條登山路線其難易程度
與天氣狀況皆不相同

→ 各條路線分別建模

→ 使用DNN (深度神經網路)演算法

模型建立

Step 1. 將各路線的資料拆分成獨立的data frame

Step 2. 將資料切分成訓練及測試集

```
train_test_split(np.asarray(temp.iloc[:,1:-2]), np.asarray(temp['SuccessPercentage']),
```

Step 3. 建立DNN模型

取6個參數做為輸入變數X：
包括嘗試次數、平均電壓、平均溫度、
平均濕度、平均風速與風向、平均輻射量

```
model = Sequential()
model.add(Dense(12, activation='relu', input_shape=(6,)))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

參數調整

以資料量最多的路線其模型為主，進行參數調整，原始模型準確率：0.46

1. 增加model units

```
model = Sequential()
model.add(Dense(24, activation='relu', input_shape=(6,)))
model.add(Dense(12, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

準確率變化：0.46→0.43

2. 改變activation function

```
model = Sequential()
model.add(Dense(12, activation='sigmoid', input_shape=(6,)))
model.add(Dense(8, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))
```

準確率變化：0.46→0.44

參數調整+模型訓練

3. 增加一層hidden layer

```
model = Sequential()
model.add(Dense(12, activation='relu', input_shape=(6,)))
model.add(Dense(8, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

準確率變化：0.46→0.45

參數調整後模型預測準確率不但沒有增加，反而還些微下降，因此維持原本模型的參數設定。

- **模型訓練：**

在訓練過程中，設定每一模型fitting均跑5個epoch

```
model.fit(x_tr, y_tr, epochs=5, batch_size=1, verbose=1)
```

模型效度驗證

- 取得測試集準確率：

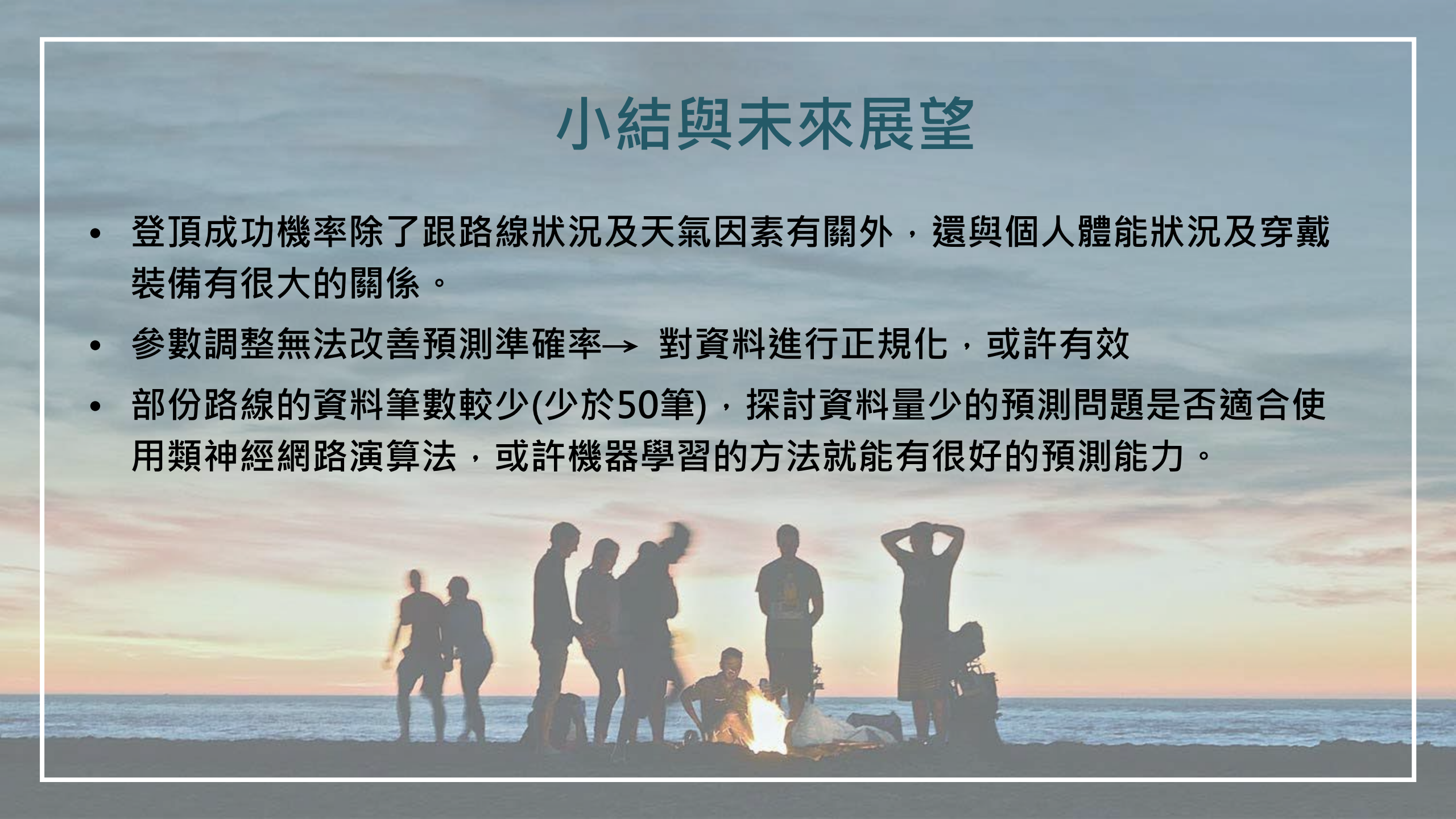
```
score = model.evaluate(x_tst, y_tst)
print('loss:',score[0] ,'accuracy', score[1])
```

- 整體平均預測準確率達63%

	loss	accuracy
df_disappointmentcleaver	0.987529	0.455598
df_emmons_winthrop	0.764610	0.440678
df_fuhrersfinger	1.971632	0.714286
df_gibralterledges	0.740012	0.833333
df_ingrahamdirect	1.140211	0.800000
df_kautzcleaver	41.850819	0.500000
df_kautzglacier	0.907372	0.300000
df_libertyringrahamdirectge	3.866420	0.500000
df_littletahoma	0.757317	0.625000
df_ptarmiganringrahamdirectge	10.142046	0.500000
df_glacieronly_nosummitattempt	0.154129	1.000000

小結與未來展望

- 登頂成功機率除了跟路線狀況及天氣因素有關外，還與個人體能狀況及穿戴裝備有很大的關係。
- 參數調整無法改善預測準確率→ 對資料進行正規化，或許有效
- 部份路線的資料筆數較少(少於50筆)，探討資料量少的預測問題是否適合使用類神經網路演算法，或許機器學習的方法就能有很好的預測能力。



THANK YOU