

Kaggle Titanic 練習

| 資訊系大四 F74092269 陳冠廷

練習0. 資料前處理

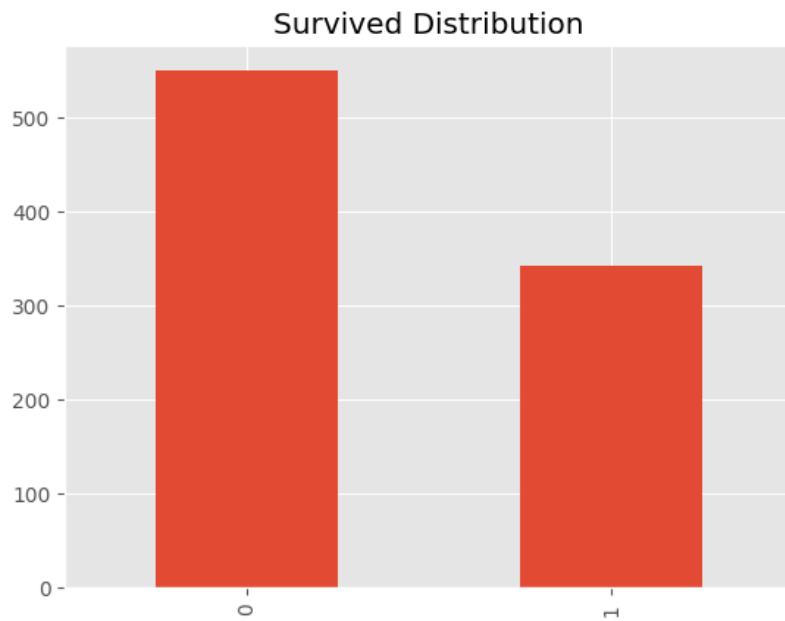
資料集分析

1. Titanic 資料集總共有 11 種欄位，如下表呈現：

欄位名稱	大略介紹	資料類型
PassengerId	辨識不同乘客的一個ID	整數值（連續）
Survived	任務主要的預測標籤（0 表示死亡，1 表示存活下來）	整數值（類別）
Pclass	分三等人（1, 2, 3），1的社會地位最高，3最低	整數值（類別）
Name	乘客姓名	字串
Sex	性別分為 Male 和 Female 兩種	字串（類別）
Age	一般的年紀數值	小數值（連續）
Sibsp	船上的兄弟姐妹/配偶數量	整數值（連續）
Parch	船上的父母/小孩數量	整數值（連續）
Ticket	票名	字串
Fare	票價	小數值（連續）
Embarked	登船的地點（S, C, Q）	字串（類別）

2. 觀察標籤的分布是否過度不平均

由下圖可以發現：死亡和存活的比例約略是 2:1，算是一點點的不平均，所以後續調參數的時候也會考慮到這點。



3. 查看資料是否含有缺失值以及佔比多少，總共有三個欄位有缺失值，佔該欄位的比例如下：

欄位名稱	缺失比例
Age	0.198653
Cabin	0.771044
Embarked	0.002245

缺失值與特徵工程

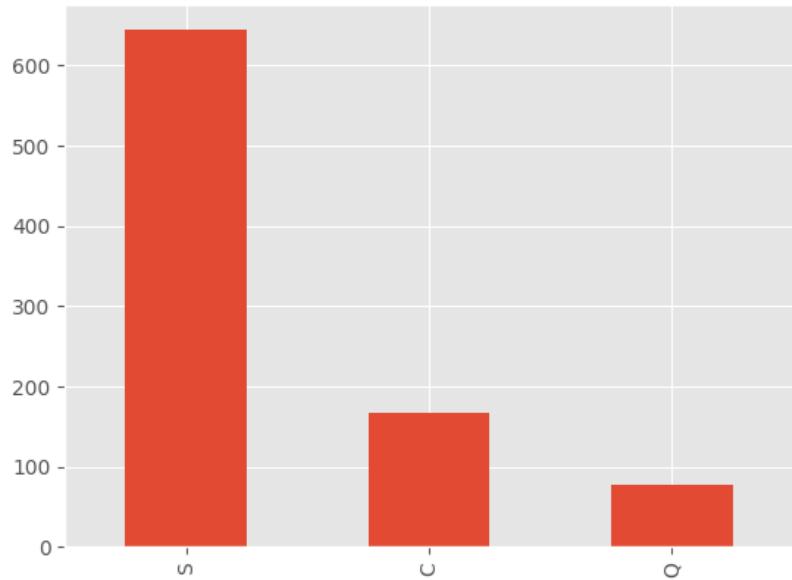
已知有三個欄位有缺失值後，我發現 `Cabin` 欄位有超過 3/4 都是缺失值，直覺來看是沒有填補的必要，因此我剛整個欄位移除。接著，我根據兩個有缺失欄位進行分析。

Embarked 缺失值

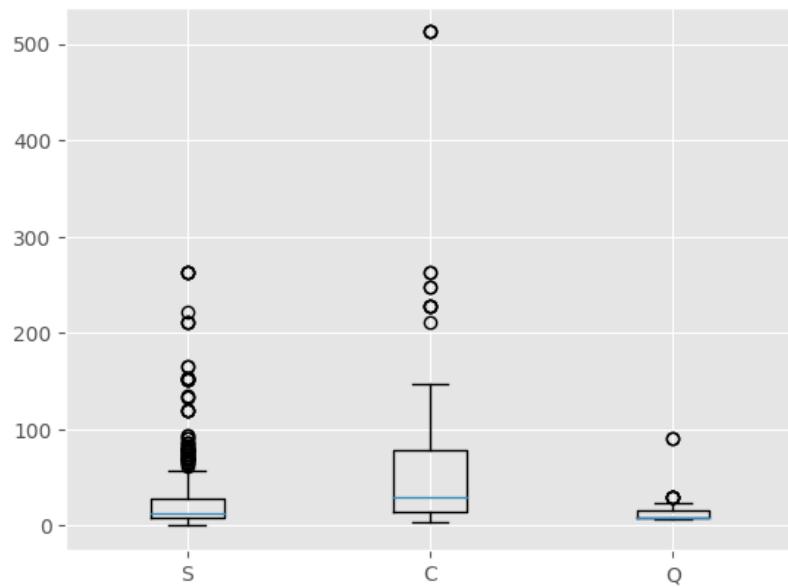
由於 `Embarked` 欄位缺失數量非常少（僅兩筆），因此我採用case study的方式去看資料。下表可以看出兩筆資料相對來說較相近，幾乎所有欄位數值都相同。

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
62	1	1	Icard, Miss. Amelie	female	38.0	0
830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0

1. 第一步我看 `Embarked` 欄位的分布，發現大致上都是'S'，其次'C'，最後才是'Q'。由分布可能會認為填'S'是比較正確的，但是接下來我做了更細緻的分析。



2. 我觀測到這些資料的 `Fare` 欄位為 `80`，在票價中幾乎是非常高的存在了。於是我想對票價做了分析，認為這或許是幫助填值的相關欄位。
- 我假設票價與 `Embarked` 是有關的，愈接近發車處通常到終點的票價愈貴，我依據 `Fare` 和 `Embarked` 繪製了下面的盒子圖。非常明顯可以看出'S'基本上不會出現80元的票價，而'C'又比'S'更有可能出現80元票價。

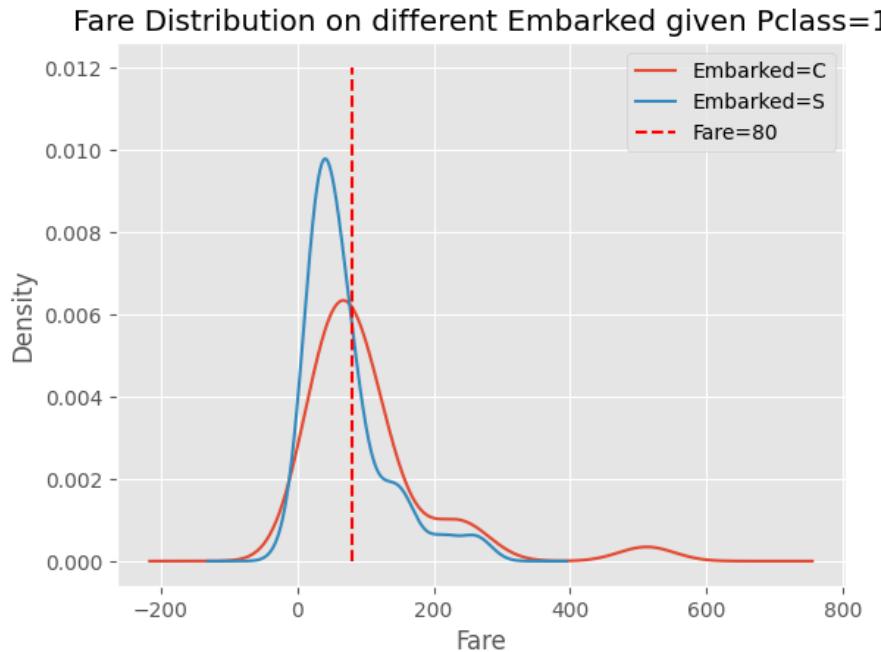


- 我考慮到票價和 `Pclass` 必有非常高的相依關係，我統計了不同 `Pclass` 的最高票價，也很值得看到只有社會地位最高的人才有可能會出現票價80的情況。

```
titanic_df.groupby("Pclass")["Fare"].agg(max)
```

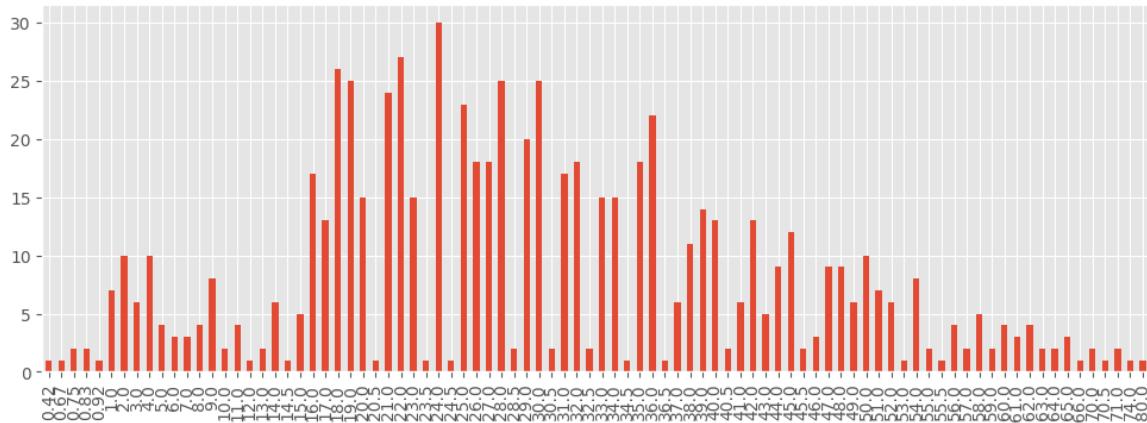
Pclass	最高票價
1	512.3292
2	73.5000
3	69.5500

- c. 只統計Embarked=['S', 'C']且Pclass為1的這些人，票價的分布曲線。從圖，可以看出80的右邊在'C'有更高的佔比（約莫有50%），因此這兩筆資料填入'C'是更好的選擇。



Age 缺失值

- 我一樣先繪製欄位分布，發現分布沒有甚麼特別的。

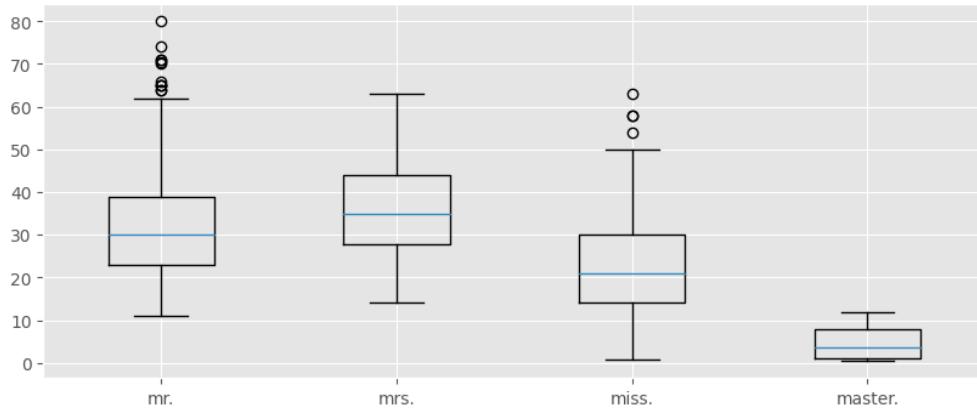


- 我發現 Name 這個欄位特別神奇，有Mr, Mrs這類的稱號，而這通常涵蓋一點年紀的意義。例如：Mrs通常已婚，年紀會較高。於是統計所有出現的字（轉小寫）並過濾掉出現少於20次的，得到以下結果：

```
{'mr.': 517,
'mrs.': 125,
'john': 44,
'miss.': 182,
'william': 62,
'henry': 33,
'james': 24,
'master.': 40,
'charles': 23,
```

```
'george': 22,
'thomas': 21}
```

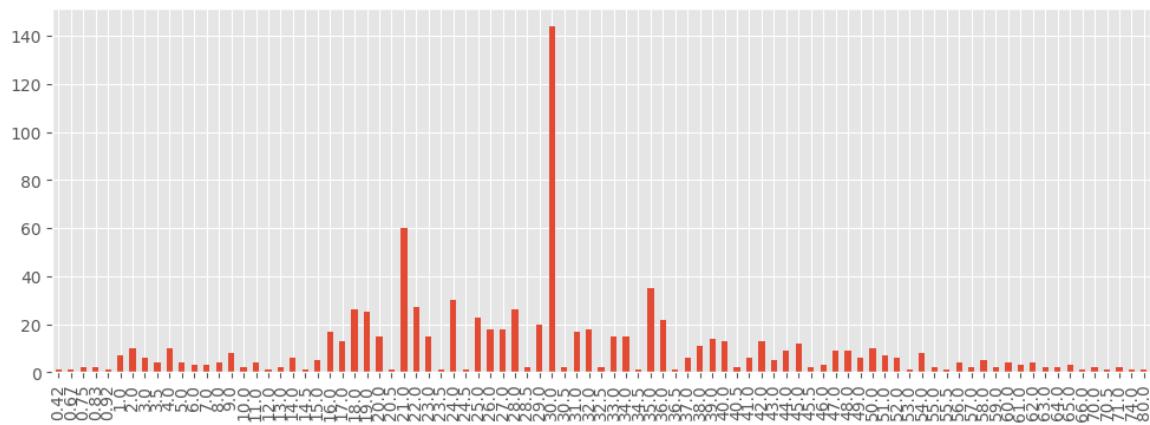
3. 其中，我認為只有 `mr.`, `mrs.`, `miss.`, `master.` 涵蓋年紀意涵，於是針對這四個做統計分析。可以更明確看到 `master.` 就是小孩的稱號，而 `miss.` 相對 `mrs.` 更年輕一點，`mr.` 則是一個最廣的分布。



4. 其中缺失值只有一筆不涵蓋這些稱號，所以該筆以所有資料Age的中位數填補。剩餘的Age我則是根據其稱號對應到的中位數去填補，如表所示：

稱號	缺失值補值
mr.	30.0
mrs.	35.0
miss.	21.0
master.	3.5
找不到	28.0

5. 填補後的分布如下圖。由於 `mr.` 的缺失非常多所以在 30.0 的地方有非常高的分布



篩選特徵

我採用直觀的篩選方法

- 我移除掉偏向 id 類型的資料：`PassengerId`, `Ticket`, `Name`
- 這些偏向親子關係的特徵 `SibSp`, `Parch` 直覺上無關聯，另一方面，我也不懂這個特徵在幹嘛...
- 以下為清理後資料的範例，只剩下5個特徵與1個標籤

Survived	Pclass	Sex	Age	Fare	Embarked
0	3	male	22.0	7.25	'S'
1	1	female	38.0	71.2833	'S'

特徵數值化

由於 Sex 和 Embarked 是非數值資料，我利用LabelEncoder將其轉為數值

練習1. 決策樹

由於在微調過程時，有去查看特徵重要性，發現Embarked的重要性常常很低，因此我在此report的多數結果都是只用4個特徵來預測(Embarked除外)。

預設參數

預測參數訓練的結果只有0.709，比助教的結果還要差。原因就出在決策樹預設參數傾向於**把資料細分到無法拆分或是沒有特徵可以繼續切割為止**，而這會出現overfitting的問題，使得在訓練資料表現非常好，但是測試的時候就很糟糕。

Data	Acc
Train	0.9831
Test	0.709

交叉驗證

由於直接使用測試資料是一種作弊的行為，因此我用交叉驗證的方式來測試參數的穩健性。我針對了三種參數做調整，考量到overfitting與前面提到的類別可能不均衡兩種問題來改善。用grid_search的方式來找尋最好的參數組合。

```
best parameters: {'class_weight': None, 'max_depth': 13, 'min_samples_leaf': 5}
best scores: 0.8370696400625979
```

最終決策樹

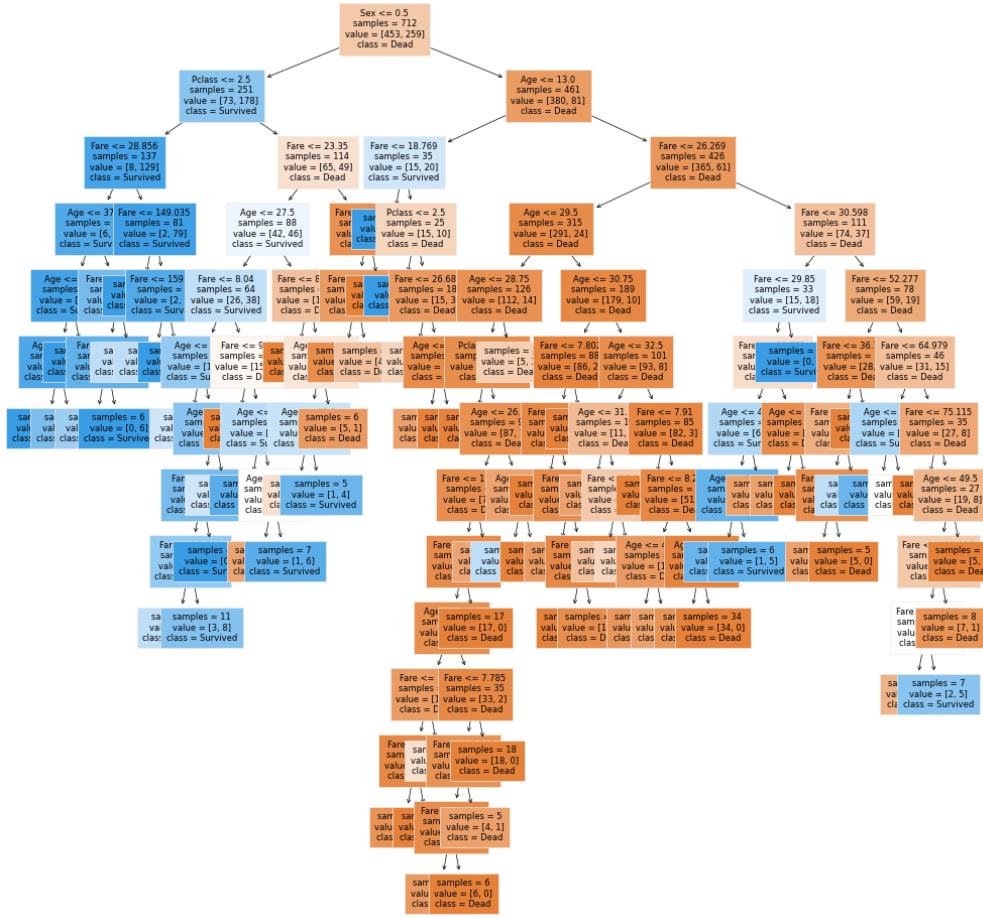
最後以交叉驗證的參數訓練決策樹於整個切分後的訓練資料，並預測測試資料後得到以下結果。比助教的baseline多了約莫 **0.07** 的分數提升。

Data	Acc
Train	0.8975
Test	0.7989

了解決策樹

我簡單的視覺化決策樹的結果

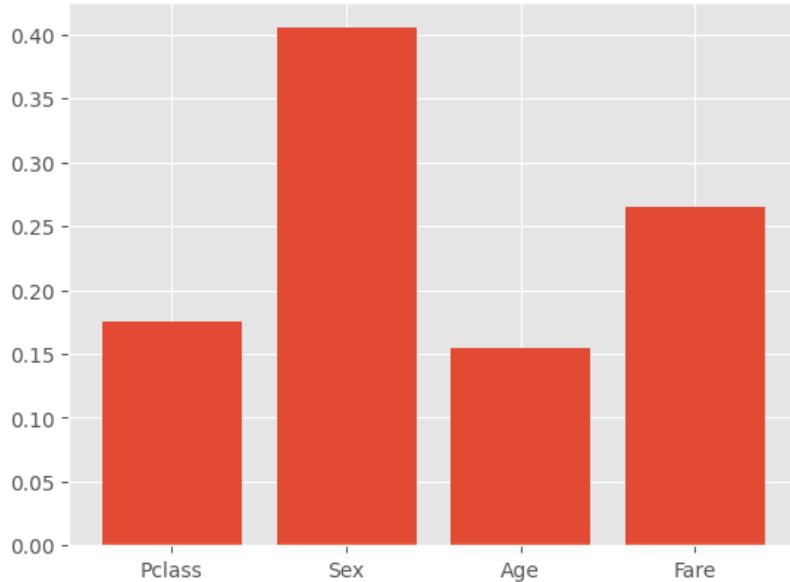
- 整棵樹視覺化 (第一步先分類性別，接著男生再依社會地位切分，而女生則是用年紀區分，一直做下去...)。從前幾步可以發現最有效的切割是非常合乎邏輯的，逃生時確實是以婦女與孩童為優先，而對於男生來說大概就是有錢得才能先活命了。



- 特徵的重要性

- 由以性別特別重要
- 這裡沒有Embarked是因為在調參時發現其重要性非常低，所以將其移除了

Feature Importance of Decision Tree



練習2. 模型方法比較

- 我使用了 random-forest 和 random-forest + logistic regression 作為模型
- random-forest 是一種集成方法透過結合多個決策樹來做到更好的分類
 - bagging的想法 ⇒ 隨機抽樣訓練資料來訓練每一個決策樹
 - 主要比決策樹可以多設定一個額外參數 `n_estimators`，決定要長幾棵樹
- random-forest+lr是取自推薦系統的方法，希望可以有效地用在這個任務上
 - 每個資料點透過決策樹的葉節點可以被分成一類，而多個決策樹就可以將一個資料點轉換到多維度的向量（葉節點ID構成的onehot表示）
 - 可以使用random-forest產生的onehot表示用logistic regression來分類
- 同於練習一的實驗方法
 - 先不調參數
 - 交叉驗證調參數
 - 測試結果

Random-Forest

- 不調參數就有決策樹調參後的水平了

Data	Acc
Train	0.9831
Test	0.7821

- 交叉驗證的最佳參數組
- best parameters: {'max_depth': 7, 'n_estimators': 60}
best scores: 0.8567488262910798

- 測試結果

Data	Acc
Train	0.9143
Test	0.8101

Random-Forest+LR

- 首先利用剛剛調好的random-forest來提取特徵，原本四個特徵可以轉換成3012維度的onehot vector。不調參數的結果非常糟，比維調後的決策樹差。

Data	Acc
Train	0.9803
Test	0.7709

- 固定剛剛的 random-forest 對於 lr 進行交叉驗證調整參數，得到以下最佳參數組
- best parameters: {'C': 0.4, 'class_weight': None}
best scores: 0.8918622848200313

- 測試結果只比沒微調提升一點，但是其在交叉驗證時卻得到最高的分數

Data	Acc
Train	0.9705
Test	0.7877

Random-Forest+PCA+LR

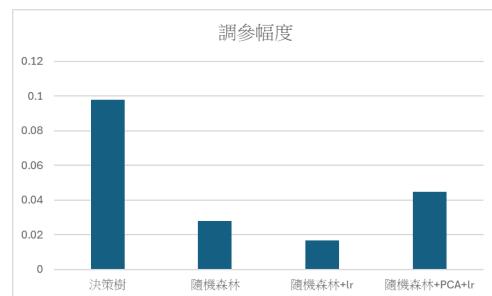
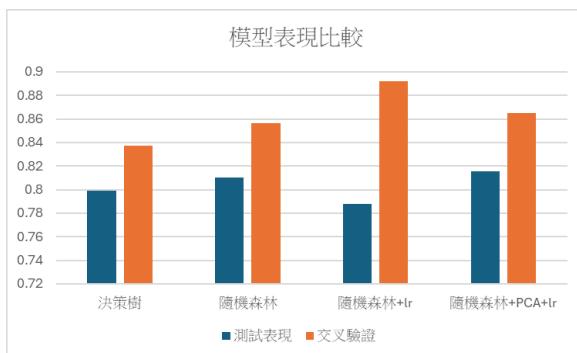
- 我認為是因為onehot的特徵維度過高導致模型表現非常糟糕，於是先使用PCA將其降維，PCA我直接將n_components制定成0.7
- 利用剛剛調好的random-forest來提取特徵，原本四個特徵可以轉換成3012維度的onehot vector，在經過PCA降維後，交由logistic regression來分類，針對 lr 進行交叉驗證調整參數，得到以下最佳參數組

```
best parameters: {'logistic_C': 0.3000000000000004, 'pca_n_components': 0.7}
best scores: 0.8652190923317683
```

- 性能比隨機森林的性能上升了一點，而且不論於交叉驗證還是測試集都達到目前最佳的測試方法

Data	Acc
Train	0.8722
Test	0.8156

最終比較結果圖



- 決策樹的表現還不錯
- 其中以隨機森林+lr的穩定性最差
- 最後是隨機森林+PCA+lr的表現最好
- 決策樹的調參提升幅度最大，其次為隨機森林+PCA+lr和隨機森林