

## GAI HW1 資訊 114 何寬羿 C34104032

### The way to improve

#### 一、輸入特徵修改

首先我檢查了各個欄位是否有缺值(null)，如下圖，可以發現"Age"、"Cabin"、"Embarked"三個欄位有缺值，且 891 筆資料中，Cabin 就有 687 筆沒有值，我認為這樣的資料即使填補後，也無法有效訓練模型，因此不予採用。

```
print(df.isna().sum())
```

PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	177
SibSp	0
Parch	0
Ticket	0
Fare	0
Cabin	687
Embarked	2

再者，"PassengerId"由於每個人皆不同，無法形成有效特徵，故也不予採用。"Name"中，包含 Mr.、Miss.、Mrs.等可嘗試判斷的重複出現的字串，不過，它與"Sex"描述的資料過於雷同，因此也不予採用。

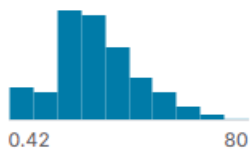
"SibSp"及"Parch"，我有思考家人的數量會影響生存率，因此嘗試將它們相加(家人數量)後去訓練模型，後來也有嘗試切割它(例如: 家人數量 < a 為 0，<= a 且 < b 為 1，<= b 為 2)來創造新的特徵，不過我發現這都無法對於準確率有有效的提升，因此後來怕干擾模型訓練，也將它們除去。

"Ticket"，我有嘗試去把這個欄位的每一個值的"數字部分"及"開頭字串部分"分別萃取出來，我發現使用"數字部分"作為特徵會比"開頭字串部分"的準確率高約 1~2%，不過即使使用數字部分作為特徵，在與其他特徵同時訓練模型時，對準確率的影響也不大，因此後來亦剔除(針對 Ticket，我也發現不同乘客的 Ticket 可能是相同的，即他們可能關係較密切，因此我透過 duplicated()嘗試找出有相同 Ticket 的人(=True)，如下圖，來作為特徵輸入，準確率也的確比"數字部分"

來的高(也是約 1~2%)，不過在與其他特徵同時訓練模型時，對準確率的影響也不大，因此後來亦剔除)。

0	A/5	21171	0
1	PC	17599	0
2	STON/O2.	3101282	0
3		113803	1
4		373450	0
..	...	...	...
886		211536	0
887		112053	0
888	W./C.	6607	1
889		111369	0
890		370376	0

(上下圖可得知，不同乘客，也能有相同 Ticket)

▲ Name	▲ Sex	# Age	▲ Ticket
<b>891</b> unique values	male 65% female 35%	 0.42 80	<b>681</b> unique values
Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	113803
Futrelle, Mr. Jacques Heath	male	37	113803

“Embarked”我做了前處理後，在與其他特徵同時訓練模型時，對準確率的影響也不大，因此後來亦剔除。

## 二、前處理

我欲採用的“Age”和“Embarked”(後來不採用)具有缺值，因此我分別使用 Median 及 Most\_Frequent 來為“Age”和“Embarked”補缺值。可以發現補值後的“Age”，對於準確度有有效的提升(我亦有使用過 mean 等參數去補值，不過還是 Median 對於準確度提升最高)。

此外，如下圖，透過 info() 可以知道我們輸入的特徵是否為數值型資料，可知部分特徵並非純數值資料，因此我有將“Sex”及“Embarked”透過 Label Encoder 轉換，用以訓練模型。

```

0  PassengerId  891 non-null  int64
1  Survived    891 non-null  int64
2  Pclass      891 non-null  int64
3  Name        891 non-null  object
4  Sex         891 non-null  object
5  Age         714 non-null  float64
6  SibSp       891 non-null  int64
7  Parch       891 non-null  int64
8  Ticket      891 non-null  object
9  Fare        891 non-null  float64
10 Cabin       204 non-null  object
11 Embarked    889 non-null  object
dtypes: float64(2), int64(5), object(5)

```

對於“Age”及“Fare”，我也將其透過計算分別的平均值和標準差，進行標準化後，才用來訓練模型。

$$\frac{X - \mu}{\sigma}$$

不同“Pclass”的生存率不同，可以透過訓練資料計算出，Pclass == 3 的乘客，存活機率較高，因此我嘗試將不同等級的艙分成(==3)和(!=3)，以“Pclass 是否為 3”的特徵去訓練模型，發現與其他特徵一起訓練下，對於準確度影響亦不大。

### 三、超參數調整

```

model = DecisionTreeClassifier(
    random_state=1012,
    max_leaf_nodes=8,
    max_depth= 15,
    criterion='gini'
)

```

在調整模型超參數之前，我發現 test acc 可以高於 83%，不過 train acc 也高達 99.57%，有嚴重的 overfitting，因此我透過嘗試不同種、不同數值的決策樹參數，來降低 overfitting，盡可能找到 test acc 夠高，overfitting 較輕微的準確度。最終結果：

```

train accuracy: 0.8314606741573034
test accuracy: 0.8156424581005587

```

成果比題目所要求的 0.7262 高出了近 10%，overfitting 也不算嚴重。只不過可能因為訓練資料較少、前處理還能有更深入的嘗試，此模型還有進步的空間。

## Different model comparison

保留與決策樹相同的特徵及前處理，我更換了以下模型，並且調整不同超參數，來嘗試更進一步提高準確率：(依 test acc 大至小排列)

### 1. Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
```

```
Randomforest = RandomForestClassifier(  
    random_state=1012,  
    n_estimators = 28,  
    criterion = 'gini',  
    min_samples_split=2,  
    max_depth=6,  
)
```

嘗試不同超參數調整，降低 overfitting(還是有些 overfitting)，且追求較高的 test acc

最終結果：

```
train accuracy: 0.8946629213483146  
test accuracy: 0.8212290502793296
```

### 2. K Neighbors Classifier

```
from sklearn.neighbors import KNeighborsClassifier
```

```
KNeighbor =KNeighborsClassifier(n_neighbors=7)
```

嘗試不同超參數調整，降低 overfitting(還是有些 overfitting)，且追求較高的 test acc

最終結果：

```
train accuracy: 0.8567415730337079  
test accuracy: 0.8156424581005587
```

以下模型的 test acc 雖低於決策樹模型，不過都有比練習 1 的 0.7262 還要高，overfitting 不算太大。

### 3. Gaussian NB

```
from sklearn.naive_bayes import GaussianNB
```

```
gNB = GaussianNB()
```

超參數能調整的空間過小，基本上對於準確度沒有提升。

最終結果：

```
train accuracy: 0.7780898876404494
test accuracy: 0.7653631284916201
```

#### 4. SVC (linear kernel)

```
from sklearn.svm import SVC
```

```
svc = SVC(kernel='linear', random_state=1012)
```

超參數能調整的空間過小，基本上對於準確度沒有提升。  
最終結果：

```
train accuracy: 0.7780898876404494
test accuracy: 0.7653631284916201
```

#### 5. Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

```
logic = LogisticRegression(random_state = 1012, max_iter=30)
```

在目前的特徵輸入下，超參數的調整，基本上對於準確度沒有特別提升。  
最終結果：

```
train accuracy: 0.797752808988764
test accuracy: 0.7430167597765364
```

部分模型在某些超參數的調整下，test acc 有比目前更高的準確度(>83%)，不過考量到為了 overfitting 的嚴重程度，因此 test acc 也跟著下降。

模型改善：

- (1) 取得更多訓練資料。目前低於 1000 筆的訓練資料，很難將準確度提高至 85%以上，甚至 90%以上。若能在訓練時增加資料量，我想模型的成效會更好。
- (2) 更細膩的資料前處理，搭配超參數調整，以提高準確度和盡量減緩 overfitting。