預測鐵達尼號旅客生存機率

第四組
楊恆軒、陳道明、杜彥陵、徐嘉澤
2019/12/05

壹、 介紹

一、 Dataset 資訊

鐵達尼號資料集為著名的數據分析競賽題目，因為其明確而有規則的特性，十分適合機器學習及深度學習的新手嘗試。

資料集提供了 1200 多名鐵達尼號乘客的相關資訊，參賽者被要求根據這些相關資訊，找出死亡與生還之旅客的特徵。

欄位變數	定義	值或特性
PassengerId	乘客ID編號	train.csv有891位 test.csv有418位 共1,309位乘客
Survived	是否生還	0 (no) / 1 (yes)
Pclass	船艙等級	1 (1st) / 2 (2nd) / 3 (3rd)
Name	姓名	包含其稱謂
Sex	性別	male / female
Age	年齡	浮點數
SibSp	在船上的兄弟姊妹和配偶人數	整數
Parch	在船上家族的父母及小孩人數	整數
Ticket	船票編號	文字
Fare	船票價格	浮點數
Cabin	船艙號碼	文字
Embarked	登船口岸	C (Cherbourg) / Q (Queenstown) / S (Southampton)

貳、 使用方法

本組以 Random Forest 及 Neural Network 為進行測試之主要方法。

Random Forest 為一種 Machine Learning 的方法；Neural Network 為 Deep Learning 的方法之一，我們期望藉由兩種方法的分析，來了解其特性及差異。

參、 Random Forest 與特徵選擇

一、 模型選擇

1. 首先找出「對抗噪聲較強」的模型：SVM、KNN、隨機森林。
2. 接著考量到「資料及對於模型的影響」：

3. 鐵達尼號的資料集非常小，僅有 891 筆資料。因此，SVM 和 KNN「遇到較大資料集，非常沒有效率」的缺點影響不大；隨機森林因其「平行化計算」的特性，在小資料集和大資料集上的運算效能都不錯。
4. 再來考量「前處理的方式」:
5. SVM 和 KNN 都是以距離的概念來進行切分或投票，前處理定義距離時較為複雜，也有處理效果不佳的風險；隨機森林則是以「不純度函數」切分樣本，毋須將資料標準化，建模較為容易。
6. 綜合考量以上幾點，我們決定以隨機森林做為本次分析的模型。

二、 套件選擇

五個主要的套件，在此次專案中被使用。分別是：

1. Numpy：提供維度陣列與矩陣的運算。
2. Pandas：提供數據操作和分析的 data frame 結構。
3. Matplotlib：提供圖形繪製的工具。
4. Seaborn：提供進階的圖表繪製，是以 matplotlib 為基礎的工具・
5. Scikit-learn：提供許多機器學習的演算法。

三、 資料集

本次 project 所使用的公開資料集，為 Kaggle 網站一個機器學習競賽所提供。

資料分為兩個檔案，train.csv 和 test.csv，競賽目標為根據提供的乘客資訊欄位，預測該名乘客是否生還，模型好壞的衡量指標為 accuracy（準確率）。

資料集中，train.csv 包含了 891 筆資料，並提供了每位乘客最終的生死結果；test.csv 則是競賽的測試資料集，418 筆、僅有乘客資訊的欄位，生死結果並未提供。

因此，此次 project 的模型好壞，僅能透過切分 train.csv 來驗證及衡量；test.csv 並沒有衡量的功能，但其乘客資訊可被拿來進行描述性統計和特徵工程分析，以利挑選合適的欄位做為模型預測乘客生死的特徵。

四、 資料欄位

欄位變數	定義	值或特性
PassengerId	乘客 ID 編號	train.csv 有 891 位 test.csv 有 418 位 共 1,309 位乘客
Survived	是否生還	0 (no) / 1 (yes)
Pclass	船票等級	1 (1^{st}) / 2 (2^{nd}) / 3 (3^{rd})
Name	姓名	包含其稱謂
Sex	性別	male / female

Age	年齡	浮點數
SibSp	在船上的兄弟姊妹和配偶人數	整數
Parch	在船上家族的父母及小孩人數	整數
Ticket	船票編號	文字
Fare	船票價格	浮點數
Cabin	船艙號碼	文字
Embarked	登船口岸	C (Cherbourg) / Q (Queenstown) / S (Southampton)

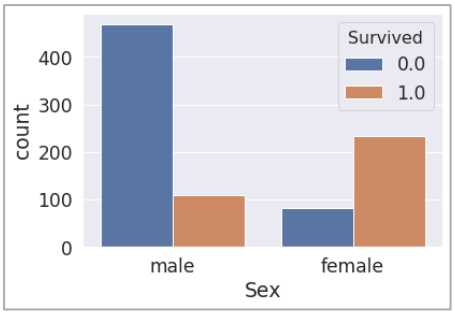
五、 特徵前處理和原始模型

- 性別與艙等

在災難發生時，通常會讓避難人員選定為老弱婦孺，加上災難資訊的傳播的第一手資訊一定是從上流下來的，因此我們選定了原始資料裡面的性別與艙等先來分析這個特徵是否影響著生存率。

1. 性別特徵：

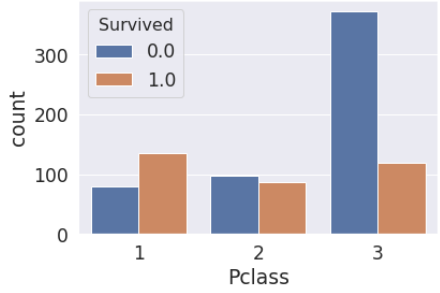
大部分的男性都死亡(僅 18.9%存活)，女性則有將近四分之三(74.2%)生還。



	Sex	Survived
0	female	0.742
1	male	0.189

2. 艙等特徵：

頭等艙的生還率較高，艙等越低的生還率越低。



	Pclass	Survived
0	1	0.630
1	2	0.473
2	3	0.242

3. 前處理：

(1) 將性別編碼：male → 0 ; female → 1

(2) 將 train.csv 切割出一部分，做為訓練模型的測試資料

(3) 定義模型的 Y：’ Survived’ ，其餘的欄位當作 X

4. 建立以「隨機森林」為方法、沒有進行其他特徵處理的原始模型：
5. Base_Model =RandomForestClassifier
   (random_state=2,n_estimators=250,min_samples_split=20,oob_score=True)

得出其測試資料(out of bag)之準確率為 0.73176

- 票價(Fare)

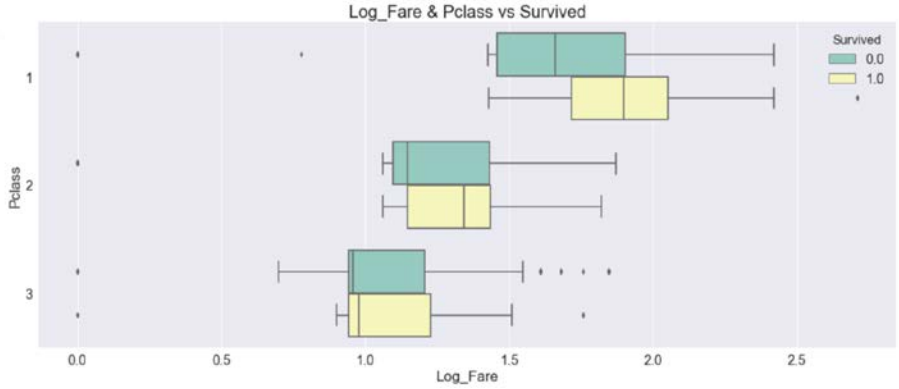
票價和艙等都是屬於彰顯乘客社會地位的一個特徵，依照主觀來判斷這是因為當危機發生後，消息的流通是會有可能最先傳到頭等艙的，故在當下情況中，先獲得資訊的人往往能把握時間逃出生天。

1. 票價計算前處理:

由於票價分布非常廣及傾斜，有非常高的票價也有非常低的，我們將票價取 log 來解決傾斜的問題，若是在 Regression Problem 中也是必要的預處理。(註 ： 取 log 之後同時也可以畫圖會好看一點)

```
# there is some bugs in log-scale of boxplot.
# alternatively, we transform x into log10(x) for visualization.
fig, ax = plt.subplots( figsize = (18,7) )
df_data['Log_Fare'] = (df_data['Fare']+1).map(lambda x : np.log10(x) if x > 0 else 0)
sns.boxplot(y='Pclass', x='Log_Fare',hue='Survived',data=df_data, orient='h'
                ,ax=ax,palette="Set3")
ax.set_title(' Log_Fare & Pclass vs Survived ',fontsize = 20)
pd.pivot_table(df_data,values = ['Fare'], index = ['Pclass'], columns= ['Survived'] ,aggfunc = 'median'
```

	Fare	
Survived	0.0	1.0
Pclass		
1	44.75	77.958
2	13.00	21.000
3	8.05	8.517



從上圖中可以看出存活下來的乘客確實平均而言付出較高的票價，我們決定測試這個特徵，然而，測試之前，我們需要將票價切分成幾個區間，才不會讓模型 overfit 的太嚴重，再來資料中含有一遺失值我們利用中位數去填補

它。

2. 切分區間:

至於切分成幾個區間這個問題，我們用極限的觀點來考慮切分區間的問題：

(1) 當切分的區間太少時，區間內的資料太多一起平均，這樣沒有辦法看出差異性，使得特徵失真。

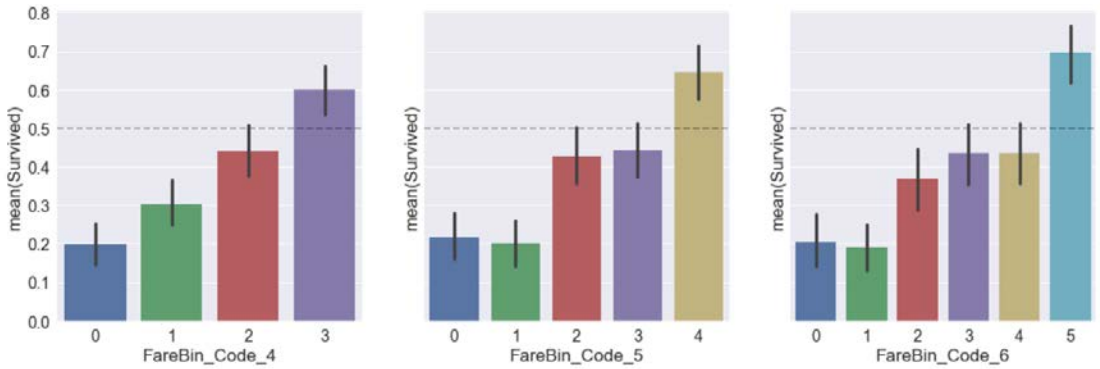
(2) 當切分區間太多時，一點點票價的不同，都影響了生存率的高低，如此一來很明顯地會 overfitting，並且，切分區間趨近於無限大時，就回到了原本的數值特徵。

```
# Making Bins
df_data['FareBin_4'] = pd.qcut(df_data['Fare'], 4)
df_data['FareBin_5'] = pd.qcut(df_data['Fare'], 5)
df_data['FareBin_6'] = pd.qcut(df_data['Fare'], 6)

label = LabelEncoder()
df_data['FareBin_Code_4'] = label.fit_transform(df_data['FareBin_4'])
df_data['FareBin_Code_5'] = label.fit_transform(df_data['FareBin_5'])
df_data['FareBin_Code_6'] = label.fit_transform(df_data['FareBin_6'])

# cross tab
df_4 = pd.crosstab(df_data['FareBin_Code_4'],df_data['Pclass'])
df_5 = pd.crosstab(df_data['FareBin_Code_5'],df_data['Pclass'])
df_6 = pd.crosstab(df_data['FareBin_Code_6'],df_data['Pclass'])

display_side_by_side(df_4,df_5,df_6)

# plots
fig, [ax1, ax2, ax3] = plt.subplots(1, 3,sharey=True)
fig.set_figwidth(18)
for axi in [ax1, ax2, ax3]:
    axi.axhline(0.5,linestyle='dashed', c='black',alpha = .3)
g1 = sns.factorplot(x='FareBin_Code_4', y="Survived", data=df_data,kind='bar',ax=ax1)
g2 = sns.factorplot(x='FareBin_Code_5', y="Survived", data=df_data,kind='bar',ax=ax2)
g3 = sns.factorplot(x='FareBin_Code_6', y="Survived", data=df_data,kind='bar',ax=ax3)
# close FacetGrid object
plt.close(g1.fig)
plt.close(g2.fig)
plt.close(g3.fig)
```

Pclass	1	2	3
FareBin_Code_4			
0	8	6	323
1	0	128	193
2	77	104	147
3	238	39	46

Pclass	1	2	3
FareBin_Code_5			
0	8	6	261
1	0	36	218
2	0	124	132
3	95	99	71
4	220	12	27

Pclass	1	2	3
FareBin_Code_6			
0	8	6	222
1	0	0	218
2	0	128	76
3	14	83	128
4	118	48	46
5	183	12	19



3. 兩件值得提的事:
   (1) Pandas 中提供了蠻多種切分數值特徵的方式，這裡選用 qcut，qcut 是以累積百分比來切分的，例如將副指令=4，就會以 0%~25%, 25%~50%, 50%~75%, 75%~100% 來切分資料，好處是可以避免某個區間內的資料過少。
   (2) 圖中虛線表示為機器隨機亂猜，應該要有 50%的準確率，如果我們的特徵工程沒辦法將各區間分離開 50%，那就沒什麼意義。
4. 切分訓練集及測試集並測試:

切分訓練集及測試集，將生還與否設為目標(Y)，其餘為訓練資料(X)，並顯示目前有的特徵

```
# splits again beacuse we just engineered new feature
df_train = df_data[:len(df_train)]
df_test = df_data[len(df_train):]
# Training set and labels
X = df_train.drop(labels=['Survived','PassengerId'],axis=1)
Y = df_train['Survived']
# show columns
X.columns
```

```
Index(['Age', 'Cabin', 'Embarked', 'Fare', 'Name', 'Parch', 'Pclass', 'Sex',
       'SibSp', 'Ticket', 'Sex_Code', 'Log_Fare', 'FareBin_4', 'FareBin_5',
       'FareBin_6', 'FareBin_Code_4', 'FareBin_Code_5', 'FareBin_Code_6'],
      dtype='object')
```

對於特徵選擇(Feature Slection)的問題，這裡我們利用前向選擇法(RFE)做特徵選擇。

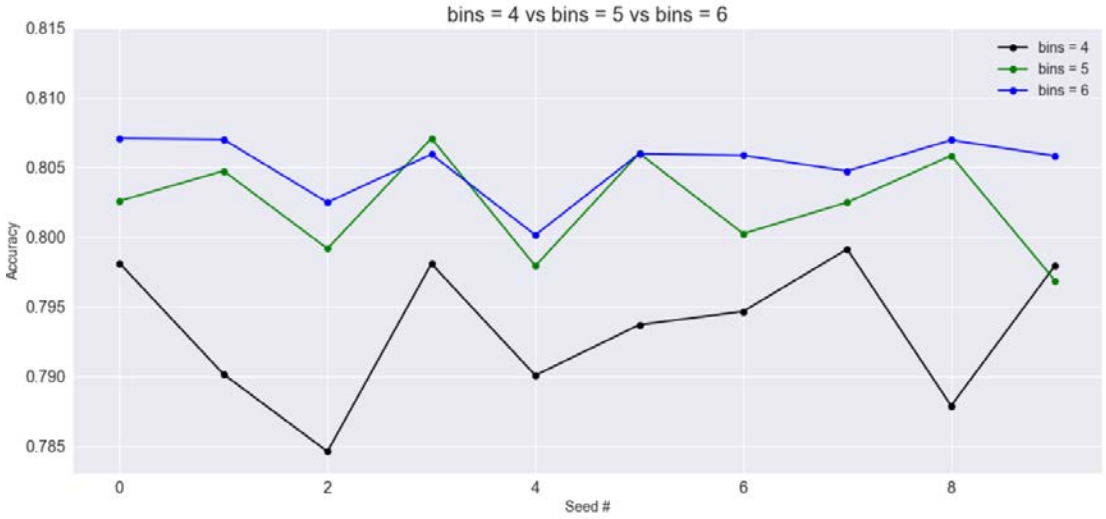
```
compare = ['Sex_Code','Pclass','FareBin_Code_4','FareBin_Code_5','FareBin_Code_6']
selector = RFECV(RandomForestClassifier(n_estimators=250,min_samples_split=20),cv=10,n_jobs=-1)
selector.fit(X[compare], Y)
print(selector.support_)
print(selector.ranking_)
print(selector.grid_scores_*100)
```

```
[ True  True  True  True  True]
[1 1 1 1 1]
[78.66981614 77.33398593 79.0144138  79.68865623 80.14308819]
```

在 CV 上我們可以看到切分成 6 份可以得到比較高的 CV 分數，但是還沒有考慮到模型的 random_state 以及 Cross-Validation 切分的方式，我們必須小心謹慎的確認切成 6 份是否真的是最好的，下面針對 CV 及模型的 random_state 進行實驗。

```
score_b4,score_b5, score_b6 = [], [], []
seeds = 10
for i in range(seeds):
    diff_cv = StratifiedKFold(n_splits=10,shuffle=True,random_state=i)
    selector = RFECV(RandomForestClassifier(random_state=i,n_estimators=250,min_samples_split=20),cv=diff_cv,n_jobs=-1)
    selector.fit(X[compare], Y)
    score_b4.append(selector.grid_scores_[2])
    score_b5.append(selector.grid_scores_[3])
    score_b6.append(selector.grid_scores_[4])
```

將結果以圖表呈現



由上圖我們可以看出切分成 4 份的準確率較低，6 份比 5 份稍微好一點，我們直接顯示準確率。

```
b4, b5, b6 = ['Sex_Code', 'Pclass','FareBin_Code_4'], ['Sex_Code','Pclass','FareBin_Code_5'],\
['Sex_Code','Pclass','FareBin_Code_6']
b4_Model = RandomForestClassifier(random_state=2,n_estimators=250,min_samples_split=20,oob_score=True)
b4_Model.fit(X[b4], Y)
b5_Model = RandomForestClassifier(random_state=2,n_estimators=250,min_samples_split=20,oob_score=True)
b5_Model.fit(X[b5], Y)
b6_Model = RandomForestClassifier(random_state=2,n_estimators=250,min_samples_split=20,oob_score=True)
b6_Model.fit(X[b6], Y)
print('b4 oob score :%.5f' %(b4_Model.oob_score_))
print('b5 oob score :%.5f '%(b5_Model.oob_score_))
print('b6 oob score : %.5f' %(b6_Model.oob_score_))
```

```
b4 oob score :0.80584
b5 oob score :0.81033
b6 oob score : 0.80135
```

最終我們可以透過票價此一因素切成 5 份得到測試資料(out of bag)之準確率從 0.73176 提高為 0.81033。

- **連結(Connected _Survival)**

這個特徵相當有意思，主要是發現了乘客持有相同的船票意味著他們可能是家人或是朋友，而在訓練集上這些互相有連結的人常常是一起活下來或是一起喪命，我們將從票根的特徵 Ticket 開始看起。

1. 連結前處理:

在發生意外時，家人朋友常常互相幫助，雖然資料上的兄弟姊妹數(SibSp)和父母小孩數(Parch)它們與一個人是否存活沒有直接關係，但是通過和票根(Ticket)比對，可以發現他是否獨自在這艘船上，而這對於他的生存機率也有很大的影響。

```
[ ]  df_train['Ticket'].describe()
```

```
count      891
unique     681
top       1601
freq         7
Name: Ticket, dtype: object
```

在 891 個票根資訊中，獨立的有 681 項，這表示一定有乘客是持有相同的票根，這意味著他們可能一起分享某一區的座位，因此我們將建立一個新的特徵家庭人數特徵(Family_size)，將兄弟姊妹數(SibSp)+父母小孩數(Parch)+1(他/她自己)方便接下來的觀察。

```
[ ]  # Family_size
     df_data['Family_size'] = df_data['SibSp'] + df_data['Parch'] + 1
```

接著建立持有相同票根的 DataFrame，並顯示姓名、票價、艙位、家庭人數。

```
deplicate_ticket = []
for tk in df_data.Ticket.unique():
    tem = df_data.loc[df_data.Ticket == tk, 'Fare']
    #print(tem.count())
    if tem.count() > 1:
        #print(df_data.loc[df_data.Ticket == tk,['Name','Ticket','Fare']])
        deplicate_ticket.append(df_data.loc[df_data.Ticket == tk,['Name','Ticket','Fare','Cabin','Family_size','Survived']])
deplicate_ticket = pd.concat(deplicate_ticket)
deplicate_ticket.head(14)
```

	Name	Ticket	Fare	Cabin	Family_size	Survived
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	PC 17599	71.2833	C85	2	1.0
234	Cumings, Mr. John Bradley	PC 17599	71.2833	C85	2	NaN
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	113803	53.1000	C123	2	1.0
137	Futrelle, Mr. Jacques Heath	113803	53.1000	C123	2	0.0
6	McCarthy, Mr. Timothy J	17463	51.8625	E46	1	0.0
146	Hilliard, Mr. Herbert Henry	17463	51.8625	E46	1	NaN
7	Palsson, Master. Gosta Leonard	349909	21.0750	NaN	5	0.0
24	Palsson, Miss. Torborg Danira	349909	21.0750	NaN	5	0.0
374	Palsson, Miss. Stina Viola	349909	21.0750	NaN	5	0.0
567	Palsson, Mrs. Nils (Alma Cornelia Berglund)	349909	21.0750	NaN	5	0.0
389	Palsson, Master. Paul Folke	349909	21.0750	NaN	5	NaN
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	347742	11.1333	NaN	3	1.0
172	Johnson, Miss. Eleanor Ileen	347742	11.1333	NaN	3	1.0
869	Johnson, Master. Harold Theodor	347742	11.1333	NaN	3	1.0

從上圖的表格觀察到:

(1) 就編號 7,24,374,567,389 這個群組來看，該家族有可能是全部一起喪命的(縱使有一個 test 的資料)，也可以從姓名中看出，5 名成員皆為 Palsson 家族，一位先生(Mr.)及兩位小姐(Miss)帶著兩位小男孩(Master)搭上了鐵達尼號，票根皆為 349909，甚至票價也是同樣的。

(2) 接著再看到編號 8,172,869 群組，皆為 Johnson 家族的成員，兩位女性(Mrs.及 Miss)帶著一位小男孩(Master)搭上了船，這則是一個三位乘客皆存活的例子。

(3) 也未必所有的群組都是同生同死，例如編號 3,137。

(4) 最後，我們可以從編號 6,146 的這個群組看出兩位一起搭船，但並非是親屬關係(姓名中的姓氏不同)，因此可以推定可能是朋友或是基於甚麼原因共同搭船的人，同樣也有可能再傳難發生時互相幫忙。

我們也可以透過家庭成員人數這個特徵來分類，Family_size = 1 但是又在群組內(即有人跟他/她持有相同票根)的，即非親屬關係，我們歸類為朋友；Family_size > 1 則為家人。

```
df_fri = deplicate_ticket.loc[(deplicate_ticket.Family_size == 1) & (deplicate_ticket.Survived.notnull())].head(7)
df_fami = deplicate_ticket.loc[(deplicate_ticket.Family_size > 1) & (deplicate_ticket.Survived.notnull())].head(7)
display(df_fri,df_fami)
print('people keep the same ticket: %.0f '%len(deplicate_ticket))
print('friends: %.0f '%len(deplicate_ticket[deplicate_ticket.Family_size == 1]))
print('families: %.0f '%len(deplicate_ticket[deplicate_ticket.Family_size > 1]))
```

	Name	Ticket	Fare	Cabin	Family_size	Survived
6	McCarthy, Mr. Timothy J	17463	51.8625	E46	1	0.0
20	Fynney, Mr. Joseph J	239865	26.0000	NaN	1	0.0
791	Gaskell, Mr. Alfred	239865	26.0000	NaN	1	0.0
195	Lurette, Miss. Elise	PC 17569	146.5208	B80	1	1.0
681	Hassab, Mr. Hammad	PC 17572	76.7292	D49	1	1.0
61	Icard, Miss. Amelie	113572	80.0000	B28	1	1.0
829	Stone, Mrs. George Nelson (Martha Evelyn)	113572	80.0000	B28	1	1.0

	Name	Ticket	Fare	Cabin	Family_size	Survived
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	PC 17599	71.2833	C85	2	1.0
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	113803	53.1000	C123	2	1.0
137	Futrelle, Mr. Jacques Heath	113803	53.1000	C123	2	0.0
7	Palsson, Master. Gosta Leonard	349909	21.0750	NaN	5	0.0
24	Palsson, Miss. Torborg Danira	349909	21.0750	NaN	5	0.0
374	Palsson, Miss. Stina Viola	349909	21.0750	NaN	5	0.0
567	Palsson, Mrs. Nils (Alma Cornelia Berglund)	349909	21.0750	NaN	5	0.0

```
people keep the same ticket: 596
friends: 127
families: 469
```

2. 創建新特徵:

觀察數據有約莫 600 位乘客和他人持有相同票根，其中大概有 75%為家庭出遊，接著依照觀察來創建一個新的特徵。

	Name	Ticket	Fare	Cabin	Family_size	Survived
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	PC 17599	71.2833	C85	2	1.0
234	Cumings, Mr. John Bradley	PC 17599	71.2833	C85	2	NaN
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	113803	53.1000	C123	2	1.0
137	Futrelle, Mr. Jacques Heath	113803	53.1000	C123	2	0.0
6	McCarthy, Mr. Timothy J	17463	51.8625	E46	1	0.0
146	Hilliard, Mr. Herbert Henry	17463	51.8625	E46	1	NaN
7	Palsson, Master. Gosta Leonard	349909	21.0750	NaN	5	0.0
24	Palsson, Miss. Torborg Danira	349909	21.0750	NaN	5	0.0
374	Palsson, Miss. Stina Viola	349909	21.0750	NaN	5	0.0
567	Palsson, Mrs. Nils (Alma Cornelia Berglund)	349909	21.0750	NaN	5	0.0
389	Palsson, Master. Paul Folke	349909	21.0750	NaN	5	NaN
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	347742	11.1333	NaN	3	1.0
172	Johnson, Miss. Eleanor Ileen	347742	11.1333	NaN	3	1.0
869	Johnson, Master. Harold Theodor	347742	11.1333	NaN	3	1.0

接著我們想知道這些擁有連結的群組的生存率為多少，因此可以看到上表的 PC 17599 這個 Ticket 群組的 Connected_Surival = 1，

其他歸類為 Connected_Surival =0.5

(1) 先將所有人的 Connected_Surival 都設定為 0.5 :

df_data[‘Connected_Surival’] = 0.5

(2) 若群組中至少有一人生還則定義

Connected_Survival = 1 : if (smax == 1.0):

```
# the same ticket family or friends
df_data['Connected_Survival'] = 0.5 # default
for _, df_grp in df_data.groupby('Ticket'):
    if (len(df_grp) > 1):
        for ind, row in df_grp.iterrows():
            smax = df_grp.drop(ind)['Survived'].max()
            smin = df_grp.drop(ind)['Survived'].min()
            passID = row['PassengerId']
            if (smax == 1.0):
                df_data.loc[df_data['PassengerId'] == passID, 'Connected_Survival'] = 1
#print
print('people keep the same ticket: %.0f '%len(deplicate_ticket))
print("people have connected information : %.0f"
      %(df_data[df_data['Connected_Survival']!=0.5].shape[0]))
df_data.groupby('Connected_Survival')[['Survived']].mean().round(3)
```

```
people keep the same ticket: 596
people have connected information : 294
```

Connected_Survival	Survived
0.5	0.283
1.0	0.728

我們得到在 596 位持有彼此持有相同票根的乘客，其中有 294 位含有連結關係，再將其分組分別計算生還率，可以看到連結=1 的生存率更是從 0.283 直接飆升至 0.728。

3. 切分訓練集及測試集並測試:

切割訓練集及測試集，並分離出生還與否(Y)以及訓練資料(X)

```
df_train = df_data[:len(df_train)]
df_test = df_data[len(df_train):]
# Training set and labels
X = df_train.drop(labels=['Survived','PassengerId'],axis=1)
Y = df_train['Survived']
```

最後加入模型、訓練、觀察 oob score。

```
connect = ['Sex_Code','Pclass','FareBin_Code_5','Connected_Survival']
connect_Model = RandomForestClassifier(random_state=2,n_estimators=250,min_samples_split=20
                                       ,oob_score=True)
connect_Model.fit(X[connect], Y)
print('connect oob score :%.5f' %(connect_Model.oob_score_))
```

```
connect oob score :0.82043
```

最終我們可以看到測試資料(out of bag)之準確率從 0.81033 來到了 0.82043，這表示我們加入了一個相當有效果的特徵。

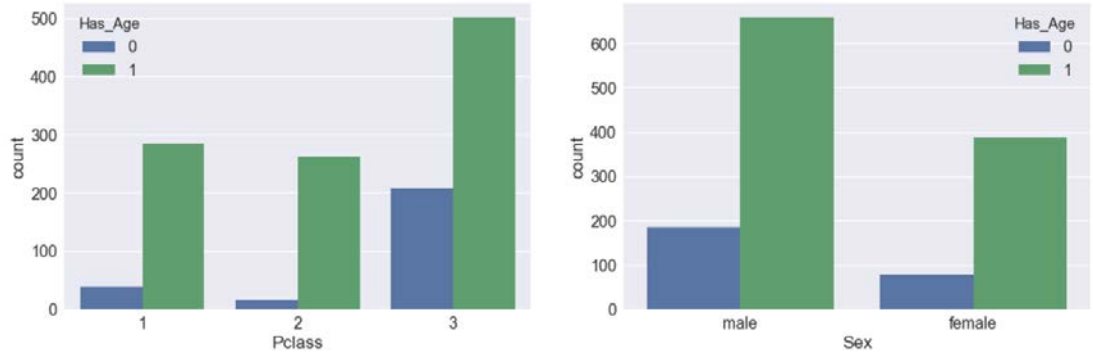
- **年齡(Age)**

在年齡特徵中我們會面臨將近 20%缺失值的問題，這很有可能影響到我們後續的預測，故我們將分兩個部分來討論:

1. 由於使用性別及艙等可以達到 0.73176 的準確率，因此如果缺失年齡大多屬於某個性別，或是屬於某個艙等，就很有可能影響預測。以下觀察缺失值分佈的情況:

```
df_data['Has_Age'] = df_data['Age'].isnull().map(lambda x : 0 if x == True else 1)
fig, [ax1, ax2] = plt.subplots(1, 2)
fig.set_figwidth(18)
ax1 = sns.countplot(df_data['Pclass'],hue=df_data['Has_Age'],ax=ax1)
ax2 = sns.countplot(df_data['Sex'],hue=df_data['Has_Age'],ax=ax2)
pd.crosstab(df_data['Has_Age'],df_data['Sex'],margins=True).round(3)
```

Sex	female	male	All
Has_Age			
0	78	185	263
1	388	658	1046
All	466	843	1309



圖左 是否缺失年齡對艙等的統計；圖右 是否缺失年齡對性別的統計

從左圖我們可以明顯的看出年齡缺失值大部分在 3 等艙，所以若年齡為重要特徵，則我們對 3 等艙的觀察就會失真，故較保守的作法為只觀察 1、2 艙等中年齡對存活與否的影響。右圖則顯示了缺失值對性別的分布，其中 466 位女性有 78 位缺失年齡(16.7%)，843 位男性有 185 位缺失年齡(21.9%)，比例差了 5%，故男性缺失年齡較多。由於 3 等艙及性別的缺失值較多，故不以此做存活率的分析。

2. 1、2 艙之中，年齡對存活與否的影響:

```
# Masks
Mask_Has_Age_P12_Survived = ( (df_data.Has_Age == 1) & (df_data.Pclass != 3 ) & (df_data.Survived == 1) )
Mask_Has_Age_P12_Dead = ( (df_data.Has_Age == 1) & (df_data.Pclass != 3 ) & (df_data.Survived == 0) )
# Plot
fig, ax = plt.subplots( figsize = (15,9) )
ax = sns.distplot(df_data.loc[Mask_Has_Age_P12_Survived, 'Age'],kde=False,bins=10,norm_hist=True,label='Survived')
ax = sns.distplot(df_data.loc[Mask_Has_Age_P12_Dead, 'Age'],kde=False,bins=10,norm_hist=True,label='Dead')
ax.legend()
ax.set_title('Age vs Survived in Pclass = 1 and  2',fontsize = 20)
```

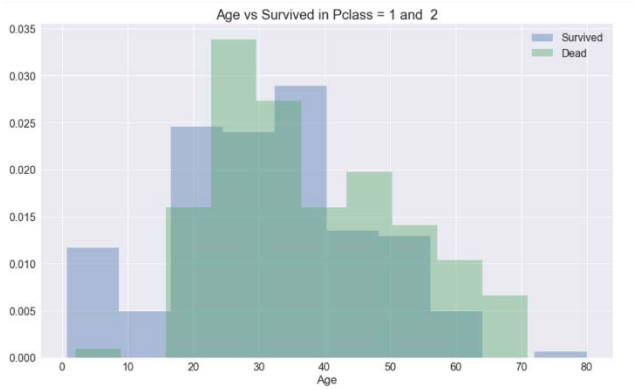


圖 1 號艙及 2 號艙當中，年齡對生還與否的影響

圖中我們可以看到，左邊藍色的部分比綠色多出一部分，也就是這部分生存率較高的，約<16 歲，表示青少年以下(包含小孩)會有較高的生存率，同時，其餘部分也顯示出了，若>16 歲，基本上年齡不算是一個顯著的特徵來判定是否生還，而 70~80 歲的這個區間，由於樣本數太少，因此不列入採計。綜合上述 3 張圖的討論，我認為找出那些<16 歲的缺失值是重要的，這會影響預測，而>16 歲的部分則不採用，否則只是擬合了噪聲，因此年齡這個特徵可以抽取出<16 歲及>16 歲做為一個 2 元特徵

填入缺失值的方式我們選擇使用姓名當中的稱謂中位數來填補，比起直接填入中位數要準確的多

```
# extracted title using name
df_data['Title'] = df_data.Name.str.extract(' ([A-Za-z]+)\.', expand=False)
df_data['Title'] = df_data['Title'].replace(['Capt', 'Col', 'Countess', 'Don',
                                             'Dr', 'Dona', 'Jonkheer',
                                              'Major','Rev','Sir'],'Rare')
df_data['Title'] = df_data['Title'].replace(['Mlle', 'Ms','Mme'],'Miss')
df_data['Title'] = df_data['Title'].replace(['Lady'],'Mrs')
df_data['Title'] = df_data['Title'].map({"Mr":0, "Rare" : 1, "Master" : 2,"Miss" : 3, "Mrs" : 4 })
Ti = df_data.groupby('Title')['Age'].median()
Ti
```

```
Title
0    29.0
1    47.0
2     4.0
3    22.0
4    36.0
Name: Age, dtype: float64
```

圖 列表為年齡中位數

(先生 - 29 歲，罕見稱謂 - 47 歲，小孩 - 4 歲，小姐- 22 歲， 女士 - 36 歲)

不動原始特徵 Age，將填滿年齡的特徵創建為 Ti_Age，分為<16 歲及>16 歲，命名為 Ti_Minor

```
Ti_pred = df_data.groupby('Title')['Age'].median().values
df_data['Ti_Age'] = df_data['Age']
# Filling the missing age
for i in range(0,5):
 # 0 1 2 3 4 5
    df_data.loc[(df_data.Age.isnull()) & (df_data.Title == i),'Ti_Age'] = Ti_pred[i]
df_data['Ti_Age'] = df_data['Ti_Age'].astype('int')
df_data['Ti_Minor'] = ((df_data['Ti_Age']) < 16.0) * 1
```

3. 切分訓練集及測試集並測試:

完成特徵工程，分離訓練集、測試集，分離出生還與否(Y)以及訓練資料(X)

```
# splits again beacuse we just engineered new feature
df_train = df_data[:len(df_train)]
df_test = df_data[len(df_train):]
# Training set and labels
X = df_train.drop(labels=['Survived','PassengerId'],axis=1)
Y = df_train['Survived']
```

加入模型、訓練、觀察 oob score

```
minor = ['Sex_Code','Pclass','FareBin_Code_5','Connected_Survival','Ti_Minor']
minor_Model = RandomForestClassifier(random_state=2,n_estimators=250,min_samples_split=20,oob_score=True)
minor_Model.fit(X[minor], Y)
print('minor oob score :%.5f' %(minor_Model.oob_score_))
```

```
minor oob score :0.84175
```

最後可以看到我們的測試資料(out of bag)之準確率從 0.82043 提升為 0.8417。

六、 Random Forest 的績效

我們以辨識生存率的準確率為指標，依序將票價(Fare)、連接(Connected _Survival)及年齡(Age)因素納入考量。發現在經過數據的處理後，其特徵值皆能提高上一階段之準確率。其中我們也將家庭人數當作特徵值，但由於效果不顯著，故不將其納入我們最後的因素中。所以我們最終以五個因素「Sex_Code」、「Pclass」、「FareBin_Code_5」「Connected_Survival」、「Ti_Minor」為建立 Model 及測試準確率的因素。

肆、 類神經網路分析(Neural Network, NN)

我們從 RF 的分析中挑選出「Sex_Code」、「Pclass」、「FareBin_Code_5」「Connected_Survival」、「Ti_Minor」為主要分析之因素，故本組以此為基礎建構的 NN Model ，如下圖:

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Activation

# Initialising the NN
model = Sequential()

# layers
model.add(Dense(9, kernel_initializer = 'uniform', activation = 'relu', input_dim = 5))
model.add(Dense(9, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(5, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(1, kernel_initializer = 'uniform', activation = 'sigmoid'))

# summary
model.summary()
```

其 Model 結構如下圖

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 9)                 54
_________________________________________________________________
dense_2 (Dense)              (None, 9)                 90
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 50
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 6
=================================================================
Total params: 200
Trainable params: 200
Non-trainable params: 0
```

其測試結果如下:

```
# Compiling the NN
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# Train the NN
model.fit(X[minor], Y, batch_size = 32, epochs = 200)
```

```
Epoch 200/200
891/891 [==============================] - 0s 45us/step - loss: 0.4218 - acc: 0.8148
```

得知準確率為 0.8148。

伍、 結果與討論及未來研究之方向

透過上方兩種方法的分析可知，當我們將 Random Forest 所得出之關鍵因素放入 Neural Network 中會發現，即使使用相同的因素， 但是利用 NN 進行深度學習後，準確率卻沒有 RF 來的高(RF 準確率:0.8417；NN 準確率:0.8148)，我們認為可能是 Titanic 的樣本數太少，所以不適用於深度學習的模型中，因為深度學習適合在大樣本數下進行，才能展現其優勢。因此我們認為，在此資料集中，RF 的表現優於 NN 是可以預期的。

從本次的 Project 中，我們發現適當的選擇考慮因素及資料的前處理對於後續的建立 Model 及辨識準確率極為重要。所以我們認為後續可以透過 tune 參數、改用其他演算法(例如，XGBoost)或是以其他 Dataset(例如，世越號資料)來測試看看，以此來改善及延伸本次的研究內容。

陸、 參考資料

1. Titanic: Machine Learning from Disaster | Kaggle
   https://www.kaggle.com/c/titanic
2. [機器學習專案] Kaggle 競賽-鐵達尼號生存預測(Top 3%)
   https://medium.com/@yulongtsai/https-medium-com-yulongtsai-titanic-top3-8e64741cc11f
3. Neural Network with Keras for Kaggle's Titanic Dataset

https://github.com/liyenhsu/Neural-Network-with-Keras-for-Kaggle-Titanic-Dataset/blob/master/titanic.ipynb

4. Logistic Regression vs Random Forest Classifier
   https://www.youtube.com/watch?v=goPiwckWE9M
5. 人工智慧、機器學習與深度學習間有什麼區別?
   https://blogs.nvidia.com.tw/2016/07/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/