

鐵達尼號生存預測

國立臺北商業大學 資訊管理系 二技 二年甲班

10636008 劉家驤 10636017 陳易陞 N1066717 林欣蓉



一、摘要

發生於 1912 年 4 月的船難共奪走了超過 1500 人的性命，鐵達尼號事件可以稱為是歷史上最慘烈的船難事故。在此專題中，我們將對鐵達尼號的 891 筆乘客資料進行探索性分析，根據此資料集提供的各個資料特徵，來判斷其與存活率間的關係，接著透過 `seaborn` 和 `matplotlib.pyplot` 等方法來呈現圖表，最後使用隨機森林來預測乘客是否能倖免於這場災難。

二、研究背景與目的

究竟為甚麼這場悲劇為何會上演？人們對於此事件充滿了好奇。經過了近百年的時間和科技的演進，鐵達尼號沈船的真相也漸漸浮出檯面。我們希望能透過自身學習過的機器學習知識、技能和 Python 程式技能，分析乘客資料，進而更深入地了解此事件，並透過參與這場 Kaggle 鐵達尼號預測競賽，增進自身實作技能，並同時經由排名了解自身能力。

三、 資料集介紹與資料集來源

資料筆數共有 892 筆，資料表欄位如下表：

欄位名稱	說明	補充
PassengerId	乘客 ID 編號	
Survival	是否倖存	0 = No · 1 = Yes
Pclass	船票等級	1 = 1 _{st} · 2 = 2 _{nd} · 3 = 3 _{rd}
Name	乘客姓名	
Sex	性別	0 為女性 · 1 為男性
Age	年齡	
Sibsp	在船上同為兄弟姐妹或配偶的數目	
Parch	在船上同為家族的父母及小孩的數目	
Ticket	船票編號	
Fare	船票價格	
Cabin	船艙號碼	
Embarked	登船的口岸	C = Cherbourg Q = Queenstown, S = Southampton
	Cherbourg：瑟堡，位於法國西北的一個城鎮，屬重要軍港和商港。	
	Queenstown：科芙，位於愛爾蘭，於 1850 年更名為皇后鎮（又稱昆士敦）。	
	Southampton：南安普敦，位於陽光燦爛的英國南方海岸。	

資料來源：<https://www.kaggle.com/c/titanic/data>

生存人數及船艙等級，可知船艙等級高生存率也高

```
In [32]: sns.countplot(df_train['Pclass'], hue=df_train['Survived'])  
display(df_train[["Pclass", "Survived"]].groupby(['Pclass'], as_index=False).mean().round(3))
```

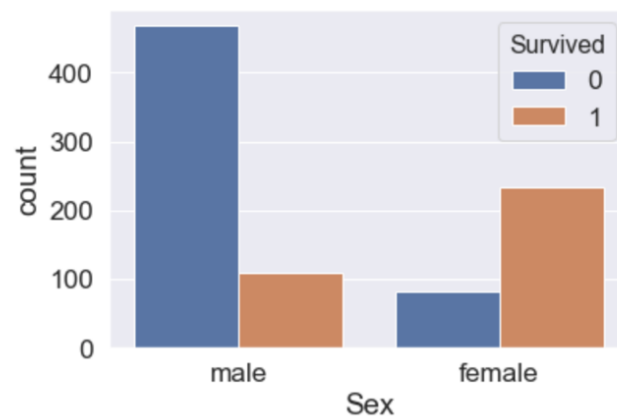
	Pclass	Survived
0	1	0.630
1	2	0.473
2	3	0.242



生存人數及性別，可知女性存活率比男性高

```
In [22]: sns.countplot(df_train['Sex'], hue=df_train['Survived'])  
display(df_train[["Sex", "Survived"]].groupby(['Sex'], as_index=False).mean().round(3))
```

	Sex	Survived
0	female	0.742
1	male	0.189



將兄弟姊妹、夫妻、父母、小孩有親屬關係的合併成一個欄位

```
In [42]: df_train['Family_size']=df_train['SibSp']+df_train['Parch']+1
```

將資料表格觀察資料

```
deduplicate_ticket=[]
for tk in df_train.Ticket.unique():
    tem=df_train.loc[df_train.Ticket == tk, 'Fare']
    #print(tem.count())
    if tem.count() > 1:
        #print(df_train.loc[df_train.Ticket == tk, ['Name','Ticket','Fare','Cabin','Family_size','Survived']])
        deduplicate_ticket.append(df_train.loc[df_train.Ticket == tk, ['Name','Ticket','Fare','Cabin','Family_size','Survived']])
deduplicate_ticket= pd.concat(deduplicate_ticket)
#deduplicate_ticket.sort_values(by='Fare').head(20)
deduplicate_ticket.head(20)
```

可以看到票根為 347742 johnson 家族兩位女性帶著一位小男孩 Master 這則是三位乘客皆存活

8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	347742	11.1333	NaN	NaN	1
172	Johnson, Miss. Eleanor Ileen	347742	11.1333	NaN	NaN	1
869	Johnson, Master. Harold Theodor	347742	11.1333	NaN	NaN	1

但是家族皆存活的案例也沒有很多，編號 137，3 票根 113803 就是一生一死

3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	113803	53.1000	C123	2	1
137	Futrelle, Mr. Jacques Heath	113803	53.1000	C123	2	0

查看資料型態發現有年齡有空值，並且在資料預處理時補上

```
In [24]: df_train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

四、 資料預處理

票價(Fare)：首先填補缺失值，由於只有一項，我們填入**中位數**

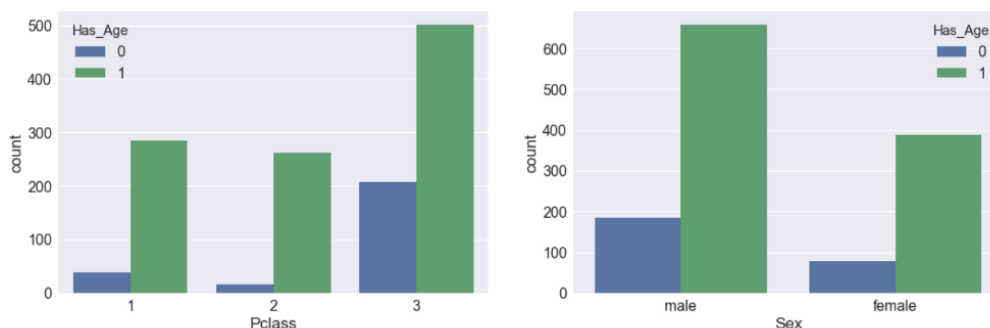
```
# Filling missing values
df_data['Fare'] = df_data['Fare'].fillna(df_data['Fare'].median())
```

年齡(Age)：在這個特徵中我們會面臨 20%缺失值的問題，和前面的票價(Fare)僅僅只有一項缺失值相比，缺的不少，而這很有可能影響預測，首先我們分兩個部分來討論：

因此如果缺失年齡特別都屬於某個性別，或是特別都屬於某個艙等，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



左：是否缺失年齡對艙等的統計，右：是否缺失年齡對性別的統計

左圖我們可以明顯的看出年齡缺失值蠻大部分在 3 等艙，如果年齡真的是個重要特徵，則我們對 3 等艙的觀察就會失真，保守的作法是觀察 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，搭配著表格看的話，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%，男性缺失年齡稍微多一點，如果年齡對生存與否有影響的話，可能可以搭配男性藉此被區分出更多的生還者(例如男性小孩生還率有可能高於男性成人)

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)
```



圖中我們可以看到，左邊藍色的部分多出了一塊，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，比起填中位數要準確的多

```
# 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
```

```
started 20:08:34 2018-06-16, finished in 29ms
```

Title	Age
0	29.0
1	47.0
2	4.0
3	22.0
4	36.0

Name: Age, dtype: float64

列表為年齡中位數，先生 - 29 歲，罕見稱謂 - 47 歲，小男孩 - 4 歲，小姐 - 22 歲，女士 - 36 歲。

五、機器學習與深度學習方法

使用隨機森林演算法來預測 Base Model

```
In [4]: # Convert Sex
df_data['Sex_Code'] = df_data['Sex'].map({'female' : 1, 'male' : 0}).astype('int')

In [5]: # split training set the testing set
df_train = df_data[:len(df_train)]
df_test = df_data[len(df_train):]

In [6]: # Inputs set and Labels
X = df_train.drop(labels=['Survived', 'PassengerId'],axis=1)
Y = df_train['Survived']

In [7]: # Show Baseline
Base = ['Sex_Code', 'Pclass']
Base_Model = RandomForestClassifier(random_state=2,n_estimators=250,min_samples_split=20,oob_score=True)
Base_Model.fit(X[Base], Y)
print('Base oob score :%.5f' %(Base_Model.oob_score_))

Base oob score :0.73176
```

使用前項選擇法(RFE)來做特徵選擇

```
In [12]: 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  78.90330269  79.46390875  80.14308819]

In [13]: 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])
```

將票價切分後使用隨機森林預測結果(比較切割成幾份分數較高)

```
In [12]: 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  78.90330269  79.46390875  80.14308819]

In [13]: 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])
```

建立家庭特徵後使用隨機森林預測結果

```
In [24]: 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
```

完成對年齡的特徵工程後使用隨機森林預測結果

```
In [31]: 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
```


六、研究結果與討論


用隨機森林完成特徵工程，分離訓練集、測試集，分離出生還與否(Y)以及訓練資料(X)，並加入模型、訓練、觀察 oob score，將結果提交至 Kaggle，成績如下。

1 submissions for jiaxiang8515		Sort by	Most recent
All Successful Selected			
Submission and Description		Public Score	Use for Final Score
submit_minor.csv 9 days ago by Jia_Xiang add submission details		0.82296	<input type="checkbox"/>
No more submissions to show			


PassengerId	Survived		
892	0	905	0
893	1	906	1
894	0	907	1
895	0	908	0
896	1	909	0
897	0	910	0
898	1	911	1
899	0	912	0
900	1	913	1
901	0	914	1
902	0	915	0
903	0	916	1
904	1	917	0
		918	1

左圖為預測結果之部分顯示

詳細請看 `submit_minor.csv` 檔案






Jia_Xiang
Joined 9 days ago · last seen in the past day


Competitions
Novice

[Home](#) [Competitions \(1\)](#) [Kernels](#) [Discussion](#) [Datasets](#) [...](#) [Edit Profile](#)

Competitions Novice




Unranked

 0  0  0

[Titanic: Machine Learning f...](#) 451st of 10411
Ongoing · Top 5%

Kernels Novice




Unranked

 0  0  0

No kernel results

Discussion Novice

Unranked

 0  0  0

No discussion results

七、 結論

Kaggle 裡有著許多資料集資訊，我們從鐵達尼號的資料集來實作，我們參考了多方論壇及資料，發現比較多人使用隨機森林這個方法，並且跟著實作後才發現到資料集有許多遺漏的資料，在做完資料分析及資料預處理後才能有效的運用特徵工程並且使用隨機森林演算法，最終將預測結果輸出。一個個小小的特徵會影響整個結果，只能說做這個 kaggle 是一項長時間又耗腦力的大工程，

八、 參考資料

<https://medium.com/@yehjames/資料分析-機器學習-第 4-1 講-kaggle 競賽-鐵達尼號生存預測-前16-排名-a8842fea7077>

[資料分析&機器學習] 第 4.1 講 : Kaggle 競賽-鐵達尼號生存預測-(前 16%排名)

<https://chtseng.wordpress.com/2017/12/24/kaggle-titanic 倖存預測-1/>

kaggle-titanic 倖存預測-1

<https://zhuanlan.zhihu.com/p/27550334>

分分钟，杀入 Kaggle TOP 5% 系列（1）