


Titanic - Machine Learning from Disaster

組員：10946012 李姍珊

10946013 趙晴


10946025 高培茵

成績：

**Titanic - Machine Learning from Disaster**493/15853
5 Submissions Left Today · OngoingTop 4%

493


分分鐘



0.80382

27

5m

**Your Best Entry!**
Your most recent submission scored 0.80382, which is an improvement of your previous score of 0.79186. Great job!Tweet this

摘要

本組選擇鐵達尼號生存預測為對這件事故有初略的了解，且是歷史上重要的事件之一。因此想透過乘客資訊像是性別、年齡...等去預估乘客是否會在鐵達尼號沉沒意外中生存下來。

介紹（研究背景及研究目的）

鐵達尼號沉沒事故是當時北大西洋發生的最大著名船難，當時與冰山擦撞前，已收到 6 次海冰警告，船行駛的速度快速，看到冰山已經為時已晚，無法及時轉向，16 個水密隔艙中的 5 個進水，而鐵達尼號的設計只能承受 4 個水密隔艙進水因此沉沒，此次災難造成 1514 人死亡。因此我們想藉由此事件，透過訓練數據分析生還人數，且能預防未來相似的事件發生。

資料集介紹(含資料特徵)及資料集來源

此競賽共有 2 份資料集，分別為 train(用來訓練模型)及 test(要求預測結果)，還有一份 data(合併 train 與 test 的資料)，以利接下來的處理。

特徵名稱	特徵定義	Key
PassengerId	乘客編號	
Survived	是否倖存	1:是 / 0:否
Pclass	船票等級	1:最高 / 2:中等 / 3:最低
Name	姓名	
Sex	性別	
Age	年齡	
SibSp	同為兄弟姐妹 或配偶的數目	
Parch	同為家族父母 及小孩的數目	
Ticket	船票編號	
Fare	船票價格	
Cabin	船艙號碼	
Embarked	登船點	C = Cherbourg Q = Queenstown S = Southampton

資料預處理

```
data.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
#   Column      Non-Null Count  Dtype
---  -
0   PassengerId  1309 non-null   int64
1   Survived     891 non-null    float64
2   Pclass       1309 non-null   int64
3   Name         1309 non-null   object
4   Sex          1309 non-null   object
5   Age          1046 non-null   float64
6   SibSp        1309 non-null   int64
7   Parch        1309 non-null   int64
8   Ticket       1309 non-null   object
9   Fare         1308 non-null   float64
10  Cabin        295 non-null    object
11  Embarked     1307 non-null   object
dtypes: float64(3), int64(4), object(5)
```

由以上合併資料結果來看，得知：

Age 缺 $1309 - 1046 = 263$ 筆資料

Fare 缺 $1309 - 1308 = 1$ 筆資料

Cabin 缺 $1309 - 295 = 1014$ 筆資料

Embarked 缺 $1309 - 1307 = 2$ 筆資料

填補缺漏值：

- Age：我們以乘客稱呼(Miss.、Ms.等...)來區分，並分別填上平均年齡。
- Fare：因只有缺 1 筆資料，所以直接用平均值填入。
- Cabin：因缺值太多，目前選擇先不作為特徵使用。
- Embarked：從分析上，發現 C 港口的乘客大多是 P1 等級的票，因此選擇填入 C 值。

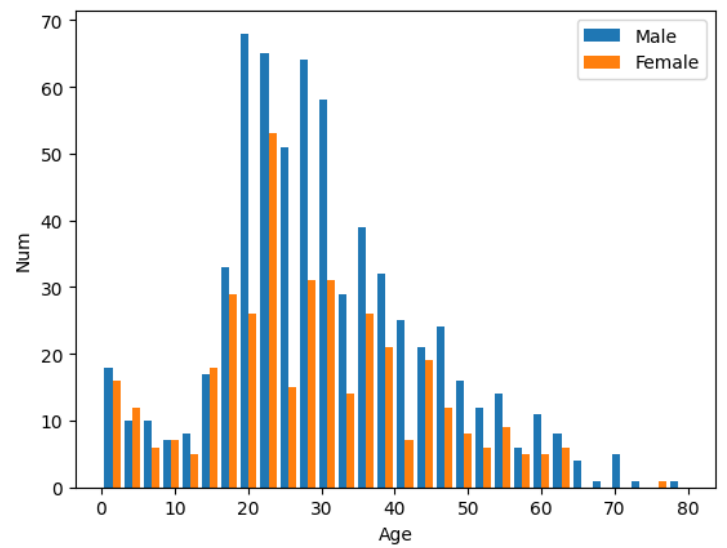
檢視非數值欄位：

Name 欄有 2 筆是重複的，而 Sex 欄只有 Male/Female 這 2 種值，其中以 Male 最多，有 843 位。

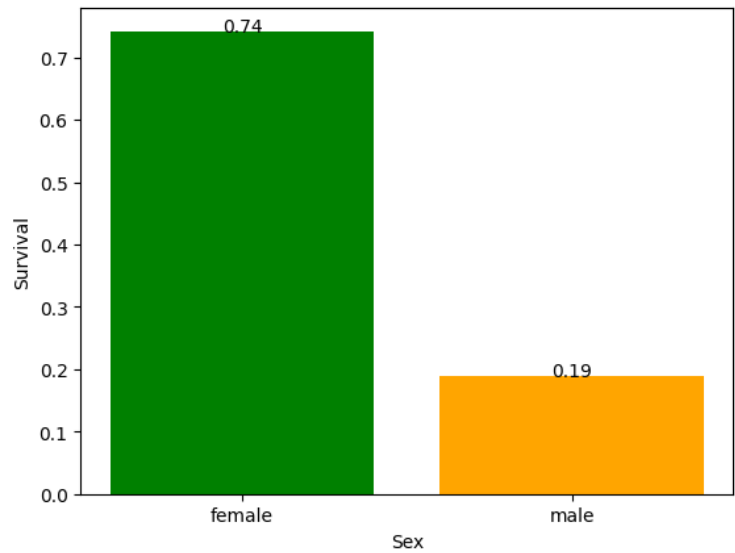
	Name	Sex	Ticket	Cabin	Embarked
count	1309	1309	1309	295	1307
unique	1307	2	929	186	3
top	Connolly, Miss. Kate	male	CA. 2343	C23 C25 C27	S
freq	2	843	11	6	914

船上的乘客各年齡層的男女比例：

由下圖可得知小於 20 歲的男女人數比例接近，但若超過 20 歲(含 20)的乘客，則男性比例則比女性多。

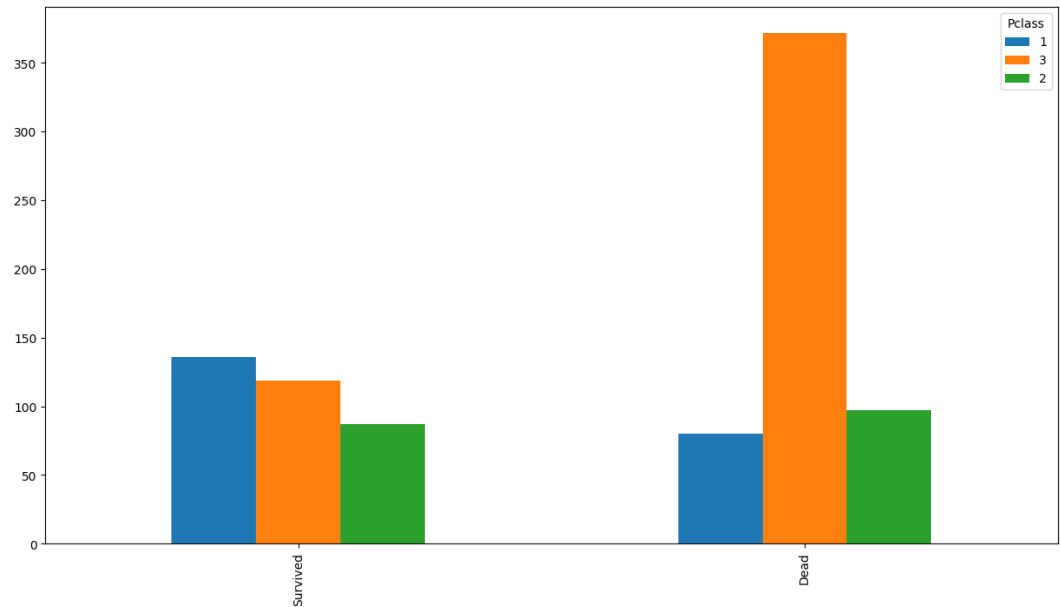


從下圖可得知，以人數來看不同性別的存活率，發現男性的存活率只有 19%，而女性則高達 74%。



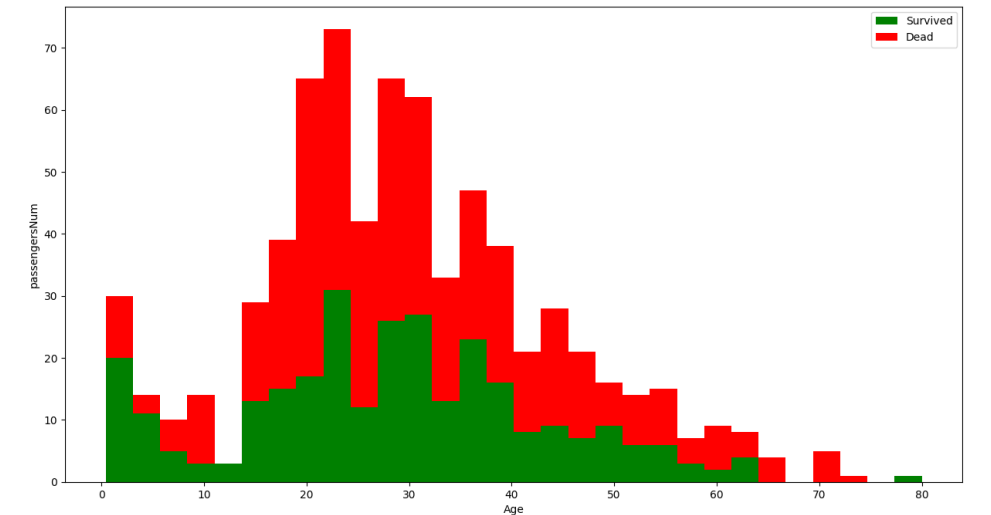
船票等級與存活的關係：

從人數來看不同船票等級的存活率：可以看出等級 3 的乘客最多，死亡者也大多為等級 3 的乘客。



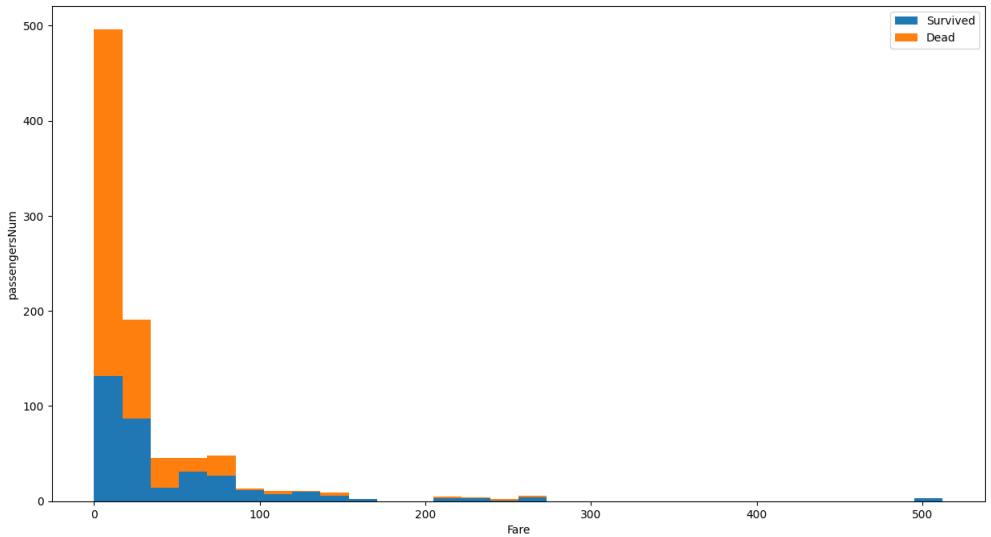
不同年齡層與倖存的關係：

從下圖可看出不同年齡層與倖存的關係，其中年齡越小存活率越高。



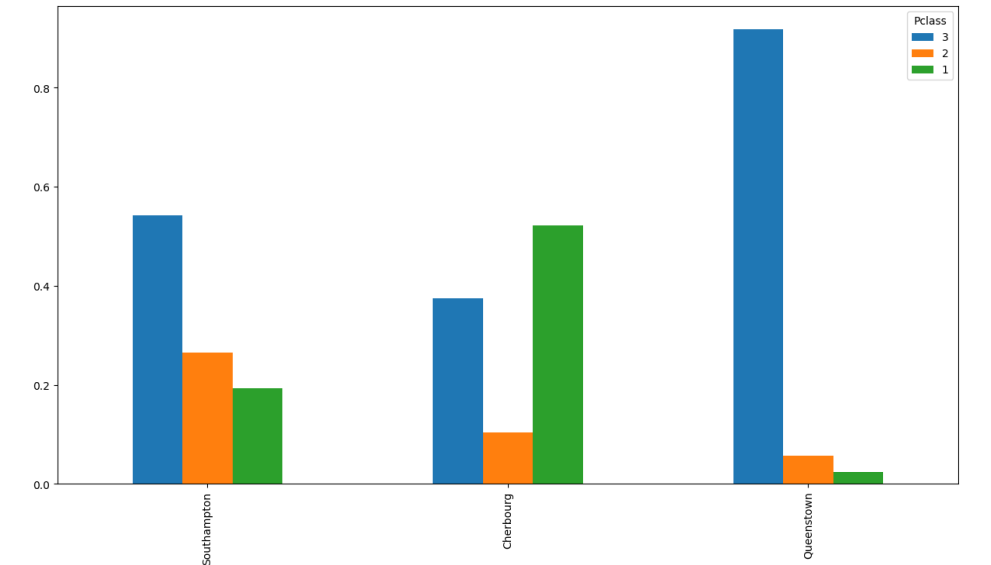
不同票價與存活的關係：

由下方結果可發現，票價愈高存活率愈大。



上岸港口與船票等級的關係：

從以下統計結果可看出不同港口的乘客有不同的經濟狀況，C 港口的乘客大多購買等級最高的票，Q 港口的乘客則是購買最低等級的票。



機器學習或深度學習方式（使用何種方式）

填補完缺漏值，確認資料無缺漏後，使用多種演算法來測試模型，並測試使用自己增加的 Feature 與使用原始 Feature 兩者的差異。最後根據測試結果，我們評估後決定使用隨機森林來訓練模型。

使用隨機森林來訓練模型：

```
modele = RandomForestClassifier(  
    n_estimators=300,min_samples_leaf=4,class_weight={0:0.745,1:0.255})  
modele.fit(X_all.values, Y_all.values)  
Y_test = modele.predict(X_test.values).astype(int)
```

將結果輸出至 gender_submission.csv 檔：

```
# 輸出結果至gender_submission.csv  
sout = pd.DataFrame({  
    "PassengerId": test_data["PassengerId"],  
    "Survived": Y_test  
})  
sout.to_csv('gender_submission.csv', index=False)
```






研究結果及討論（含模型評估與改善）

使用多種演算法測試與評估：

```
X_all = data.iloc[:891,:].drop(["PassengerId","Survived"], axis=1)  
Y_all = data.iloc[:891,:]["Survived"]  
X_test = data.iloc[891,:].drop(["PassengerId","Survived"], axis=1)  
  
logisticRegression = LogisticRegression()  
svc = SVC()  
kNeighborsClassifier = KNeighborsClassifier(n_neighbors = 3)  
decisionTreeClassifier = DecisionTreeClassifier()  
randomForestClassifier = RandomForestClassifier(  
    n_estimators=300,min_samples_leaf=4,class_weight={0:0.745,1:0.255})  
gradientBoostingClassifier = GradientBoostingClassifier(  
    n_estimators=500,learning_rate=0.03,max_depth=3)  
xGBClassifier = XGBClassifier(max_depth=3, n_estimators=300, learning_rate=0.03)  
lGBMClassifier = LGBMClassifier(max_depth=3, n_estimators=500, learning_rate=0.02)  
all = [logisticRegression, svc, kNeighborsClassifier, decisionTreeClassifier,  
    randomForestClassifier, gradientBoostingClassifier, xGBClassifier, lGBMClassifier]
```

```
邏輯斯回歸(LogisticRegression): 0.8832584269662922  
支援向量機(SVC): 0.6386267166042447  
K-近鄰演算法(KNN): 0.7991385767790261  
決策樹(DecisionTreeClassifier): 0.8417727840199751  
隨機森林(RandomForestClassifier): 0.8664794007490636  
GradientBoostingClassifier: 0.8956429463171037  
XGBClassifier: 0.8900124843945069  
LGBMClassifier: 0.8978651685393257
```

根據以上測試結果，發現 XGB、GBDT、LGBM、邏輯斯回歸、隨機森林等都有蠻高的分數，因此我們分別將分數超過 0.85 的演算法上傳至 Kaggle 評分，得出以下分數：

	XGBClassifier.zip Complete · ShanShanLi-33 · 6h ago	0.78468
	LGBMClassifier.zip Complete · ShanShanLi-33 · 7h ago	0.77751
	RandomForestClassifier.zip Complete · ShanShanLi-33 · 7h ago	0.80382
	LogisticRegression.zip Complete · ShanShanLi-33 · 7h ago	0.78947
	GradientBoostingClassifier.zip Complete · ShanShanLi-33 · 7h ago	0.78468

結論

一開始在測試階段時，我們有嘗試自己額外添加 Feature，但實際上傳評分時效果並不佳，後來使用原始 Feature 與測試多個演算法過程中，我們得出 LightGBM 是最好的，但上傳至 Kaggle 卻是 RandomForest 最好，我們目前推測可能是因為資料的過度擬和才會造成此問題，必須再對資料進行更好的處理，從而來解決問題並獲得更好的分數。

參考文獻

1. [Kaggle - Learn](#)
2. [輕量化的梯度提升機](#)
3. [機器學習中的參數調整](#)
4. [Python 資料視覺化筆記](#)
5. [隨機森林 \(Random forest\)](#)