HW 1 - First visit in Kaggle data

資訊 112 王維瀚

我所研究的 Kaggle 資料集為”Titanic – Machine Learning from Disaster”，是一個比較適合入門者學習的資料集。

這份資料包含了 PassengerId，即給予每個搭乘鐵達尼號的乘客一個編號，每個乘客分別有以下參數：Survived(1 是存活，0 是死亡)、Pclass(艙位，1 代表頭等艙，2 代表商務艙，3 代表經濟艙)、Name(乘客姓名及稱謂)、Sex、Age、Sibsp(攜帶兄弟姊妹或配偶的數量)、Parch(攜帶父母或子女的數量)、Ticket(票券的編號)、Fare(票價)、Cabin(房號)、Embarked(登船地點，C = Cherbourg, Q = Queenstown, S = Southampton)。

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Training set 裏面包含了編號 1~891 號乘客的上述資訊，test set 則有 892~1309 號乘客的部分資訊。我配合此資料集介紹裡面的引導，試圖找出各個參數與該名乘客存活與否之間的關聯性。首先觀察 train 和 test 資料集裡面的詳細資訊，train 中的 Survived 平均值為 0.38，表示前 891 名乘客中，僅有 38%的人存活。Train 和 test 中的 Pclass 平均值分別為 2.31 和 2.27，表示經濟艙的乘客應該是多於頭

等艙的。train 和 test 的乘客平均年齡分別為 29.70 和 30.27，算是相當年輕。Sibsp 和 Parch 兩項的平均在 train 和 test 裡分別為(0.52, 0.38)及(0.45, 0.39)，且兩個資料集的 Sibsp 的 50%和 Parch 的 75%都還是 0，表示攜家帶眷的人占少數，其中帶父母子女來的又比帶兄弟姊妹或配偶的來的少。最後票價的部分，train 和 test 的平均分別是 32.20 和 35.63，75%卻都只有 31.00 和 31.50，而最大值則是 512.33，表示多數人買的是平價票券，只有小於 25%的人買高於平均值的票券。從資料集本身可觀察出的資訊，大概是這些。

```
train.describe()
```

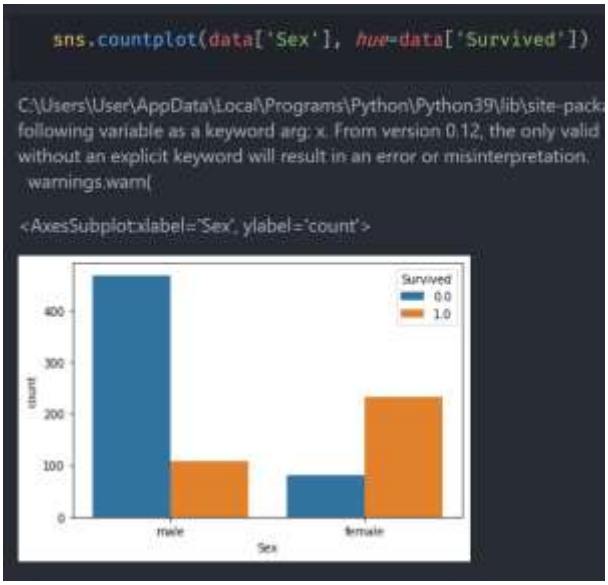
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
test.describe()
```

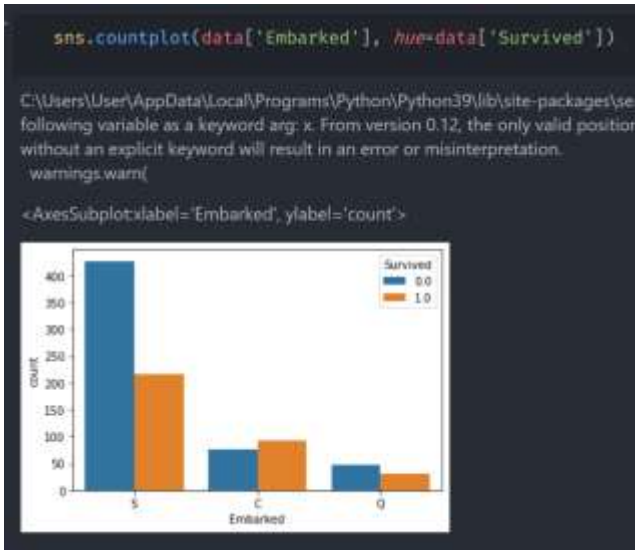
	PassengerId	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	3.000000	76.000000	8.000000	9.000000	512.329200

接著針對幾項參數與存活狀況進行比較。首先是艙位，從下頁圖左可以發現頭等艙的乘客存活比例最高，經濟艙的乘客則有很大部分都死亡了。

再來是性別與存活狀況的比較。由下頁圖右發現女性的存活率遠大於男性，可猜想如同電影情節般，先將逃生艇的位子讓給女性。



第三個是登船地點與存活狀況的比較，第一時間我們不會覺得這兩者之間有何關聯，但看到了長條圖之後，可以發現從 Southampton 登船的旅客有一大部分都死亡了。我們可以猜測從這裡登船的旅客或許都是經濟狀況相對普通，只購買便宜票券的人。因為從前面圖中已知頭等艙的乘客存活率相較其他二者為高，故這樣的猜測是有其道理的。

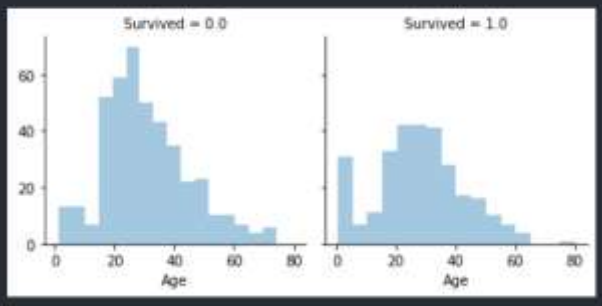


將年齡分布與存活狀況進行比較時，可由下頁圖看出，存活的分布相較死亡的分布朝年齡較小的部分偏近了一些，且接近 0 歲的幼兒

幾乎都存活了下來。可猜測逃生艇的位置除了讓給婦女外，也讓給了弱小的孩子們。

```
g = sns.FacetGrid(data, col='Survived')
g.map(sns.distplot, 'Age', kde=False)
```

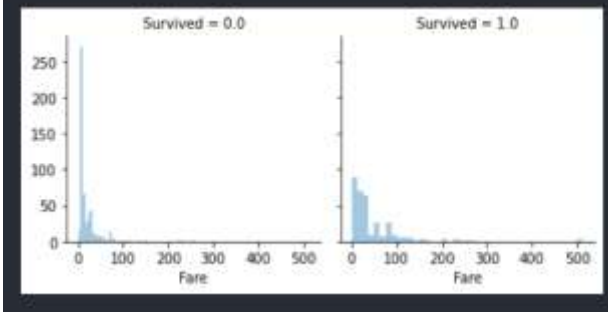
```
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\s
deprecated function and will be removed in a future version. Pl
similar flexibility) or `histplot` (an axes-level function for histogr
  warnings.warn(msg, FutureWarning)

<seaborn.axisgrid.FacetGrid at 0x22eefd6d700>
```



再觀察票價與存活狀況，發現購買低票價的旅客幾乎都死亡了，這也可以與前面的觀察相互輝映。

```
g = sns.FacetGrid(data, col='Survived')
g.map(sns.distplot, 'Fare', kde=False)
```

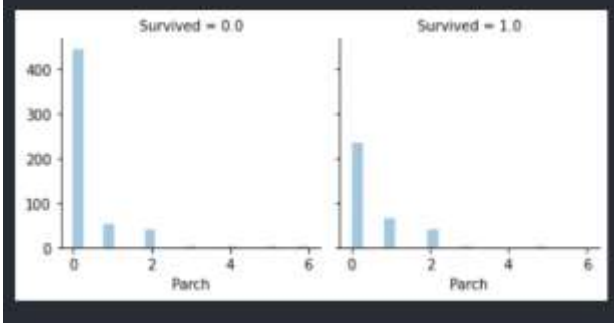
```
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\sit
deprecated function and will be removed in a future version. Ple
similar flexibility) or `histplot` (an axes-level function for histogra
  warnings.warn(msg, FutureWarning)

<seaborn.axisgrid.FacetGrid at 0x22eefea2400>
```



再來同時觀察攜家帶眷數量與存活狀況的比較，雖然獨自旅行的人本來就比較多，但從下圖仍可發現沒帶家屬來的那些人死亡率明顯較高，可推測帶了家庭後，可能可以互相照應，又或者在搭乘救生艇時，以家庭為一個單位，就比較不容易被留下來罹難。

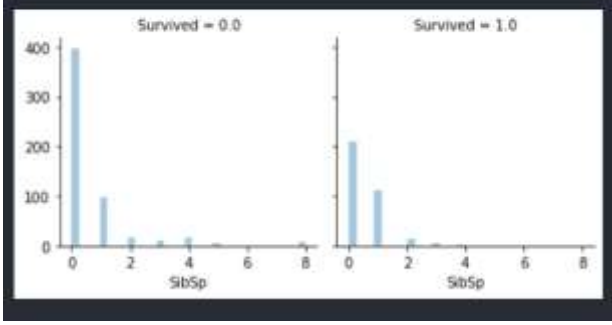
```
g = sns.FacetGrid(data, col='Survived')
g.map(sns.distplot, 'Parch', kde=False)
```

```
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\si
deprecated function and will be removed in a future version. Ple
similar flexibility) or `histplot` (an axes-level function for histogra
  warnings.warn(msg, FutureWarning)

<seaborn.axisgrid.FacetGrid at 0x22ef000f1f0>
```




```
g = sns.FacetGrid(data, col='Survived')
g.map(sns.distplot, 'SibSp', kde=False)
```

```
C:\Users\User\AppData\Local\Programs\Python\Python39\lib\si
deprecated function and will be removed in a future version. Ple
similar flexibility) or `histplot` (an axes-level function for histogra
  warnings.warn(msg, FutureWarning)

<seaborn.axisgrid.FacetGrid at 0x22ee7b9a3a0>
```



接下來運用特徵工程將一些不能直接拿來利用的資料進行處理，比如姓名這項資料對資料分析沒有直接用途，但由於其中包含了該乘客的稱謂，我們可將此稱謂提取出來，並合併那些數量過少的稱謂，只留下最大宗的 Mr, Mrs, Miss, Master。這些稱謂有其性別意義，可以幫助我們在預測 test 資料集的時候，進行更準確的判斷。

再來把票號的資訊取出前面英文的部分，因為相同的英文代碼可能代表的是房間的位置，後面的號碼沒有意義所以省略，如果只有號碼的票號就用 X 來表示

```
data['Ticket_info'] = data['Ticket'].apply(
    lambda x : x.replace(".","").replace("/","").strip().split(' ')[0]
    if not x.isdigit() else 'X')
```

接著發現 train 的 Age, Cabin, Embarked 以及 test 的 Age, Fare, Cabin 有空值的情況，為了使其不影響整體預測，我們要補上特定的值。登船港口指遺漏少數，直接以占最多的 S 填入；費用因為只遺漏了一筆，所以以票價的平均填入之。Cabin 空著的有很多，我們以 NoCabin 表示。

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 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)
memory usage: 83.7+ KB
```

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    object
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

Age 的部分則使用 Random Forest，根據其他參數來推斷空著的年齡是多少。

```
dataAgeNull = data[data["Age"].isnull()]
dataAgeNotNull = data[data["Age"].notnull()]
remove_outlier = dataAgeNotNull[(np.abs(dataAgeNotNull["Fare"]-dataAgeNotNull["Fare"].mean())>
                  (4*dataAgeNotNull["Fare"].std())) |
                   (np.abs(dataAgeNotNull["Family_Size"]-dataAgeNotNull["Family_Size"].mean())>
                   (4*dataAgeNotNull["Family_Size"].std()))
                  ]
rfModel_age = RandomForestRegressor(n_estimators=2000,random_state=42)
ageColumns = ['Embarked', 'Fare', 'Pclass', 'Sex', 'Family_Size', 'Title1', 'Title2','Cabin','Ticket_info']
rfModel_age.fit(remove_outlier[ageColumns], remove_outlier["Age"])

ageNullValues = rfModel_age.predict(X= dataAgeNull[ageColumns])
dataAgeNull.loc[:,"Age"] = ageNullValues
data = dataAgeNull.append(dataAgeNotNull)
data.reset_index(inplace=True, drop=True)
```

最後即可進行最一開始的目標：test 資料集中，乘客生存狀況的預測。

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(criterion='gini',
                            n_estimators=1000,
                            min_samples_split=12,
                            min_samples_leaf=1,
                            oob_score=True,
                            random_state=1,
                            n_jobs=-1)

rf.fit(dataTrain.iloc[:, 1:], dataTrain.iloc[:, 0])
print("%.4f" % rf.oob_score_)
```

```
0.8294
```

同時可以推算各參數對於生存狀況影響的重要性：

```
pd.concat((pd.DataFrame(dataTrain.iloc[:, 1:].columns, columns = ['variable']),
           pd.DataFrame(rf.feature_importances_, columns = ['importance'])),
          axis = 1).sort_values(by='importance', ascending = False)[:20]
```

	variable	importance
4	Sex	0.264997
2	Fare	0.163890
6	Title2	0.152698
0	Age	0.131891
3	Pclass	0.091048
5	Family_Size	0.070839
8	Cabin	0.067029
7	Ticket_info	0.031735
1	Embarked	0.025873

發現性別、票價、稱謂(Title2)、年齡影響最多。

```
rf_res =  rf.predict(dataTest)
submit['Survived'] = rf_res
submit['Survived'] = submit['Survived'].astype(int)
submit.to_csv('submit.csv', index= False)
```

最後將結果輸出至 submit.csv 檔，預覽結果如下：

submit

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
...	...	...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	1

418 rows × 2 columns

以上即為這次對鐵達尼號生存狀況研究的結果。