HW0_titanic

0813458 簡辰穎

1. Import Library

Import

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import RFECV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn.preprocessing import LabelEncoder
```

2. Load Data

將 data load 進 jupyter notebook

Load Data

```
train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')
```

3. Find Missing Data

看 data 裡面是否有缺失值的存在

```
train_data.isnull().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
dtype: int64
```

```
test_data.isnull().sum()
```

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

可以發現不論是 training data 還是 test data 在 Cabin 皆有大量的缺失值

4. 處理缺失值 & 觀察各屬性與是否存活的關係

a. Cabin: 直接刪除，因為缺失值過多

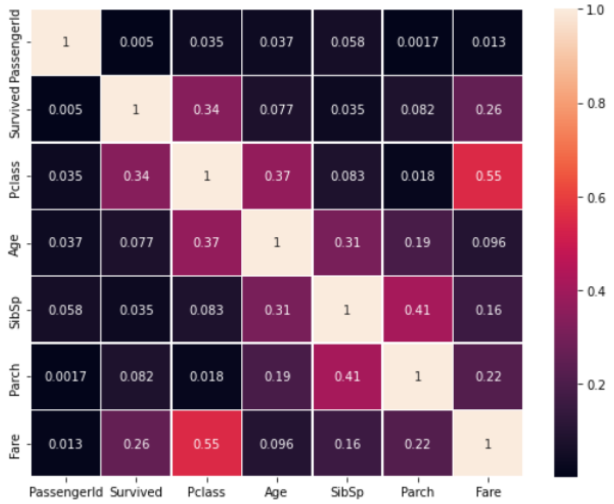
(training data 891 項中缺失 687 項，testing data 418 項中缺失 327 項)

```
train_data.drop(['Cabin'], axis=1, inplace=True)
test_data.drop(['Cabin'], axis=1, inplace=True)
```

b. 觀察各屬性與是否存活的關係

```python
fig, ax = plt.subplots(figsize=(9,7))          # Sample figsize in inches
sns.heatmap(train_data.corr().abs(), annot= True, linewidths=.5, ax = ax)
```

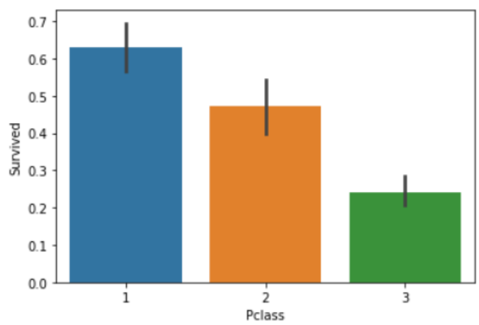
<AxesSubplot:>



Pclass

- Pclass 代表的是船票級別，可以反映出一個人的社會經濟地位
- 其中以 Pclass1 的存活率為最高: 船票級別越高生存可能性越大
- 船票級別在一定程度上也和 Fare 以及 Ticket 存在高度相關，通常船票級別越高，其價格也會越貴

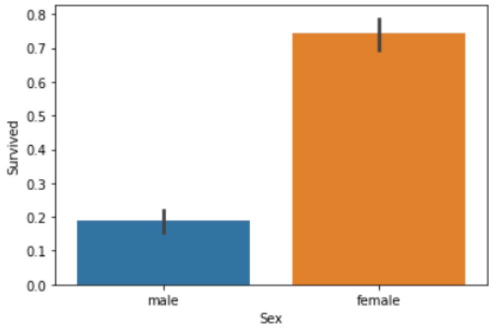
```python
sns.barplot(x='Pclass', y='Survived', data=train_data)
plt.show()
```



Sex

- 可以發現女性的生存率相較於男性多出許多
- 藉由直方圖可以輕易看出女性存活人數相比男性而言較高，而死亡人數相較男性也低出許多

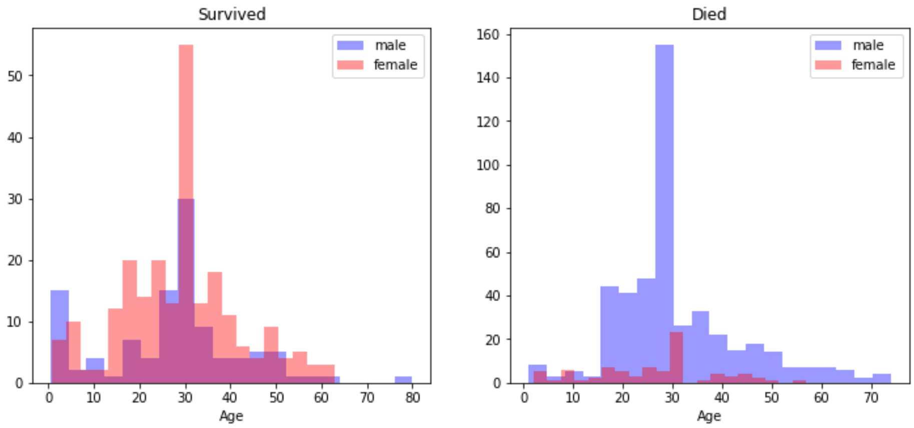
```
]: sns.barplot(x='Sex', y='Survived', data=train_data)
plt.show()
```



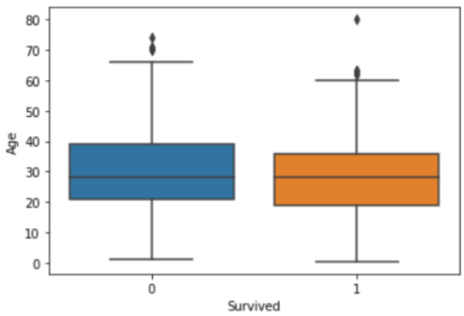
Age

- 乘客年齡主要分布在 20-40 歲
- 不論性別較年輕的乘客似乎較有機會存活

```
#比較男女年齡差異是否影響生存率 Age
male = 'male'
female = 'female'
fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(10, 3))
women = train_data[train_data['Sex']==female]
men = train_data[train_data['Sex']== male]
ax = sns.distplot(men[men['Survived']==1].Age, bins=25, label = male, ax = axes[0], kde =False, color="blue")
ax = sns.distplot(women[women['Survived']==1].Age,bins=25, label = female, ax = axes[0], kde =False, color="red")
ax.legend() #顯示圖例
ax.set_title('Survived')
ax = sns.distplot(men[men['Survived']==0].Age, bins=25, label = male, ax = axes[1], kde = False, color="blue")
ax = sns.distplot(women[women['Survived']==0].Age, bins=25, label = female, ax = axes[1], kde = False, color="red")
ax.legend() #顯示圖例
ax.set_title('Died');
```



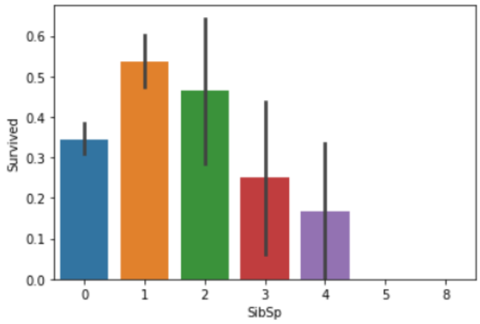

```
]: sns.boxplot(x='Survived', y='Age', data=train_data)
plt.show()
```



SibSp

- 有一個兄弟姊妹或有伴侶的人似乎存活率較高

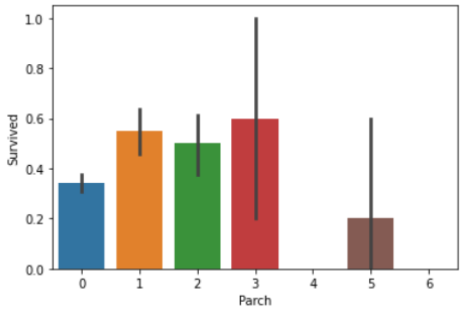
```
]: sns.barplot(x='SibSp', y='Survived', data=train_data)
plt.show()
```



Parch

- 相比並沒有和父母或孩童一起出遊的乘客而言，有的人存活率較高

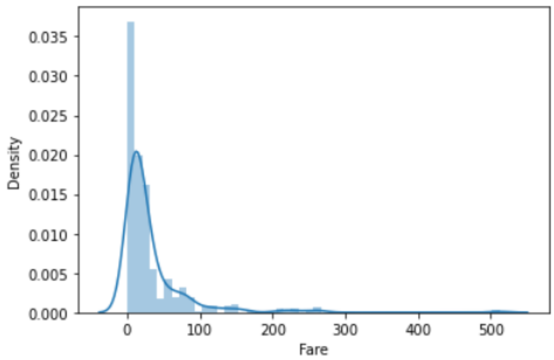
```
: sns.barplot(x='Parch', y='Survived', data=train_data)
plt.show()
```



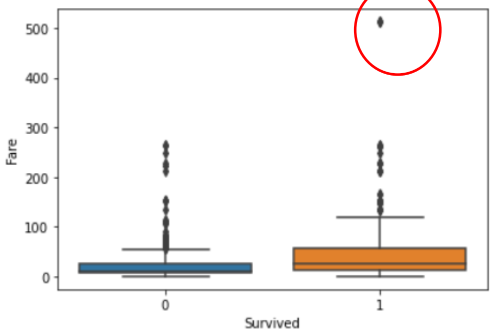
Fare

- 存活下來的乘客票價普遍較高一點
- 有 outlier(紅色圈圈處)

```
]: #Fare
sns.distplot(train_data.Fare)
plt.show()
```

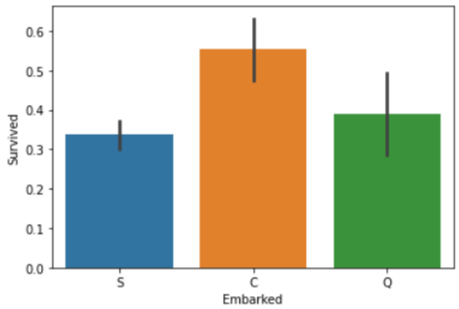


```python
sns.boxplot(x='Survived', y='Fare', data=train_data)
plt.show()
```



Embarked

```python
sns.barplot(x='Embarked', y='Survived', data=train_data)
plt.show()
```



5. 處理缺失值

Name

- 因為姓名沒有缺失值，利用姓名的稱謂特性填補其可能對應的年齡

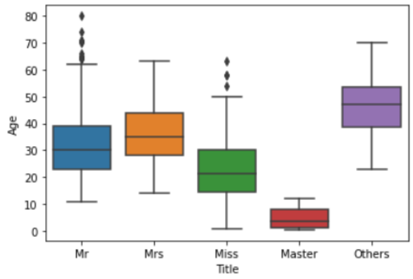
```python
whole_data = train_data.append(test_data)
whole_data['Title'] = whole_data.Name.str.extract(r'([A-Za-z]+)\.', expand=False) #正規表達式
whole_data.Title.value_counts()
```

```
Mr          757
Miss        260
Mrs         197
Master       61
Rev           8
Dr            8
Col           4
Mlle          2
Major         2
Ms            2
Jonkheer      1
Don           1
Capt          1
Lady          1
Countess      1
Mme           1
Dona          1
Sir           1
Name: Title, dtype: int64
```

```
Common_Title = ['Mr', 'Miss', 'Mrs', 'Master']
whole_data['Title'].replace(['Ms', 'Mlle', 'Mme'], 'Miss', inplace=True)
whole_data['Title'].replace(['Lady'], 'Mrs', inplace=True)
whole_data['Title'].replace(['Sir', 'Rev'], 'Mr', inplace=True)
whole_data['Title'][~whole_data.Title.isin(Common_Title)] = 'Others'
```

```
C:\Users\Administration\anaconda3\envs\Lab_2021\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

```
train_data = whole_data[:len(train_data)]
test_data = whole_data[len(train_data):]
sns.boxplot(x='Title', y='Age', data=train_data)
plt.show()
```



填補 Age 缺失值

```
AgeMedian_by_titles = train_data.groupby('Title')['Age'].median()
AgeMedian_by_titles
```

```
Title
Master      3.5
Miss       21.5
Mr         30.0
Mrs        35.0
Others     47.0
Name: Age, dtype: float64
```

```
for title in AgeMedian_by_titles.index:
    train_data['Age'][(train_data.Age.isnull()) & (train_data.Title == title)] = AgeMedian_by_titles[title]
    test_data['Age'][(test_data.Age.isnull()) & (test_data.Title == title)] = AgeMedian_by_titles[title]
```

填補 Embarked, Fare 缺失值

- Fare 有 outlier，用最大值代替 outlier

```
train_data['Embarked'].fillna(train_data.Embarked.mode()[0], inplace=True)
```

```
test_data['Fare'].fillna(test_data['Fare'].median(), inplace=True)
```

```
train_data.Fare.sort_values(ascending=False).head(5) #Fare有三個outlier
```

```
258    512.3292
737    512.3292
679    512.3292
88     263.0000
27     263.0000
Name: Fare, dtype: float64
```

```
train_data.loc[train_data.Fare>512, 'Fare'] = 263
train_data.Fare.sort_values(ascending=False).head(5)
```

```
258    263.0
88     263.0
27     263.0
341    263.0
737    263.0
Name: Fare, dtype: float64
```

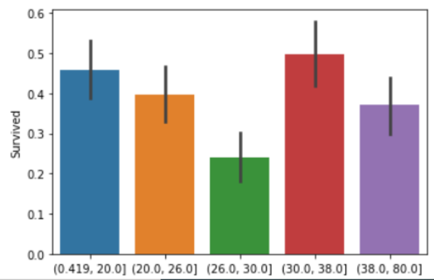
6. Date Transformation

```
train_data['Sex_Code'] = train_data['Sex'].map({'female':1, 'male':0}).astype('int')
test_data['Sex_Code'] = test_data['Sex'].map({'female':1, 'male':0}).astype('int')
```

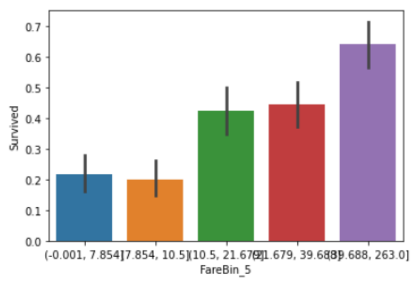
```
train_data['Embarked_Code'] = train_data['Embarked'].map({'S':0, 'C':1, 'Q':2}).astype('int')
test_data['Embarked_Code'] = test_data['Embarked'].map({'S':0, 'C':1, 'Q':2}).astype('int')
```

Age 與 Fare 皆屬與連續數值，切割成 5 個數值區間再轉換成區間的 label，較有利於模型預測結果

```
train_data['AgeBin_5'] = pd.qcut(train_data['Age'], 5)
test_data['AgeBin_5'] = pd.qcut(test_data['Age'], 5)
sns.barplot(x='AgeBin_5', y='Survived', data=train_data)
plt.show()
```




```
train_data['FareBin_5'] = pd.qcut(train_data['Fare'], 5)
test_data['FareBin_5'] = pd.qcut(test_data['Fare'], 5)

sns.barplot(x='FareBin_5', y='Survived', data=train_data)
plt.show()
```



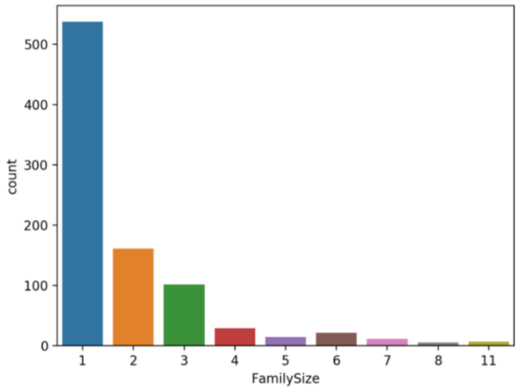

```
label = LabelEncoder()
train_data['AgeBin_Code_5'] = label.fit_transform(train_data['AgeBin_5'])
test_data['AgeBin_Code_5'] = label.fit_transform(test_data['AgeBin_5'])

label = LabelEncoder()
train_data['FareBin_Code_5'] = label.fit_transform(train_data['FareBin_5'])
test_data['FareBin_Code_5'] = label.fit_transform(test_data['FareBin_5'])
```

```
train_data['Title_Code'] = train_data.Title.map({'Mr':0, 'Miss':1, 'Mrs':2, 'Master':3,'Others':4}).astype('int')
test_data['Title_Code'] = test_data.Title.map({'Mr':0, 'Miss':1, 'Mrs':2, 'Master':3,'Others':4}).astype('int')
```

7. 建立其它屬性

FamilySize: SibSp 與 Parch 屬性皆有關於家庭人數，因此將兩個屬性合併成一個新的 FamilySize 屬性

```
train_data['FamilySize'] = train_data.SibSp + train_data.Parch + 1
test_data['FamilySize'] = test_data.SibSp + test_data.Parch + 1
sns.countplot(train_data.FamilySize)
plt.show()
```



Alone: 由於透過上方直方圖可以看出大多數乘客皆是屬於獨立旅行，對於預測結果可能無法帶來太大意義，因此建立一個是否獨立旅行的屬性

```
]: train_data['Alone'] = train_data.FamilySize.map(lambda x: 1 if x == 1 else 0)
   test_data['Alone'] = test_data.FamilySize.map(lambda x: 1 if x == 1 else 0)
   sns.countplot(train_data.Alone)
   plt.show()
```

Surname, TixPref, SurTix, IsFamily, Child, FamilyId, ConnectedSurvival

- Titanic 中倖存下來的乘客有許多皆是家庭成員，因為他們會互相幫助尋找出口
- 提取乘客姓氏+門票名稱，重複的則可能為一家人
- Connected Survival 關聯生存: 對於每個家庭而言，如果至少有一個倖存下來，我們假設其他人也能倖存下來。

```
whole_data = train_data.append(test_data)
whole_data['Surname'] = whole_data.Name.str.extract(r'([A-Za-z]+),', expand=False)
whole_data['TixPref'] = whole_data.Ticket.str.extract(r'(.*\d)', expand=False)
whole_data['SurTix'] = whole_data['Surname'] + whole_data['TixPref']
whole_data['IsFamily'] = whole_data.SurTix.duplicated(keep=False)*1
sns.countplot(whole_data.IsFamily)
plt.show()
```

```
: whole_data['Child'] = whole_data.Age.map(lambda x: 1 if x <=16 else 0)
  FamilyWithChild = whole_data[(whole_data.IsFamily==1)&(whole_data.Child==1)]['SurTix'].unique()
  len(FamilyWithChild)
: 66
```

```
: whole_data['FamilyId'] = 0
  x = 1
  for tix in FamilyWithChild:
       whole_data.loc[whole_data.SurTix==tix, ['FamilyId']] = x
       x += 1
```

```
: whole_data['ConnectedSurvival'] = 0.5
  Survived_by_FamilyId = whole_data.groupby('FamilyId').Survived.sum()
  for i in range(1, len(FamilyWithChild)+1):
      if Survived_by_FamilyId[i] >= 1:
          whole_data.loc[whole_data.FamilyId==i, ['ConnectedSurvival']] = 1
      elif Survived_by_FamilyId[i] == 0:
              whole_data.loc[whole_data.FamilyId==i, ['ConnectedSurvival']] = 0
              train_data = whole_data[:len(train_data)]
              test_data = whole_data[len(train_data):]

  sns.barplot(x='ConnectedSurvival', y='Survived', data=train_data)
  plt.show()
```

當乘客有以下屬性時通常生存率較高：

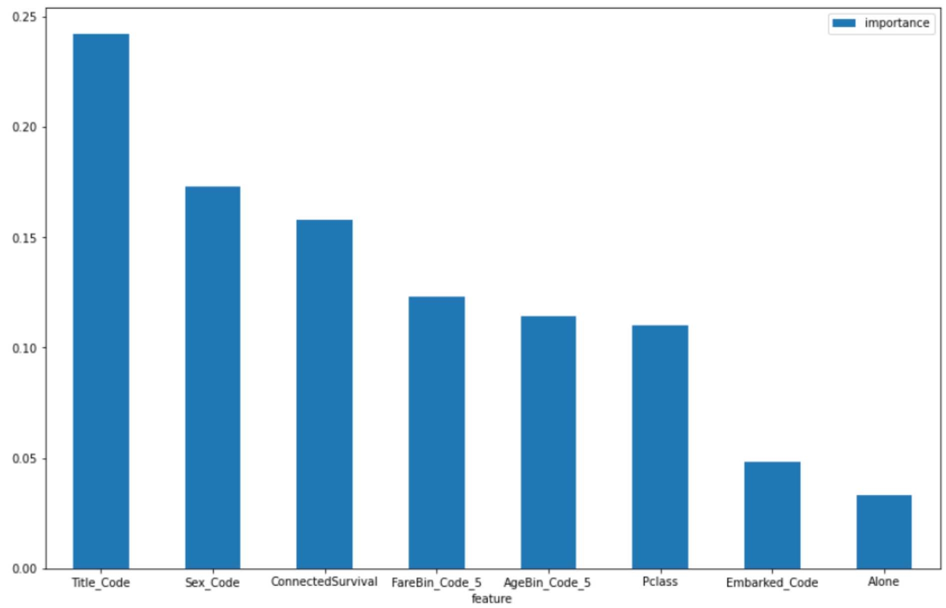
1. 與家人一起旅行
2. 家裡有 1 個或更多孩子
3. 家庭中有 1 個或更多倖存者

8. 建立模型並預測結果

```
x_train = train_data.drop(['Age', 'Embarked', 'Fare', 'Name', 'Parch', 'PassengerId', 'Sex', 'SibSp', 'Survived',
                           'Ticket', 'Title', 'AgeBin_5', 'FareBin_5', 'FamilySize', 'Surname', 'TixPref', 'SurTix', 'IsFamily',
                           'Child', 'FamilyId'], axis=1)

y_train = train_data['Survived']
```

```
model = RandomForestClassifier(n_estimators=200, random_state=2)
```

```
model.fit(x_train,y_train)

fig, ax = plt.subplots(figsize=(14,9))
importance = pd.DataFrame({'feature':x_train.columns, 'importance': np.round(model.feature_importances_,3)})
importance = importance.sort_values('importance', ascending=False).set_index('feature')
importance.plot(kind='bar', rot=0, ax = ax)
plt.show()
```



⇨ 選擇前幾項較為重要的屬性

參考網址: https://medium.com/analytics-vidhya/kaggle-titanic-survival-prediction-top-3-ea6c8dcc9b6c

```python
final = ['Title_Code', 'Sex_Code', 'ConnectedSurvival','FareBin_Code_5','AgeBin_Code_5', 'Pclass' ]
```

```python
grid_param = {
 'n_estimators': [100, 200, 300],
 'criterion':['gini', 'entropy'],
 'min_samples_split': [2, 10, 20],
 'min_samples_leaf': [1, 5],
 'bootstrap': [True, False],
}
gd_sr = GridSearchCV(estimator=model,
 param_grid=grid_param,
 scoring='accuracy',
 cv=5,
 n_jobs=-1)
gd_sr.fit(x_train[final], y_train)
best_parameters = gd_sr.best_params_
print(best_parameters)
```

```
{'bootstrap': False, 'criterion': 'gini', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 200}
```

```python
model = RandomForestClassifier(n_estimators=200, bootstrap=False, criterion= 'gini', min_samples_leaf=10,
                               min_samples_split=2, random_state=2)
```

```python
all_accuracies = cross_val_score(estimator=model, X=x_train, y=y_train, cv=5)
all_accuracies
```

```
array([0.87709497, 0.8258427 , 0.83707865, 0.80337079, 0.88764045])
```

```python
all_accuracies.mean()
```

```
0.8462055112673404
```

- 先用 GridSerchCV 看參數數值多少時效果最好

```python
x_test = test_data[final]
model.fit(x_train[final],y_train)
prediction = model.predict(x_test)
output = pd.DataFrame({'PassengerId': test_data.PassengerId, 'Survived': prediction.astype(int)})
output.to_csv('titanic.csv', index=False)
```

Name	Submitted	Wait time	Execution time	Score
titanic.csv	just now	1 seconds	0 seconds	0.80382

Complete

Jump to your position on the leaderboard

427	Chen Ying, Chien	0.80382	9	3m

Your Best Entry

Your submission scored 0.80382, which is an improvement of your previous score of 0.77990. Great job!

Tweet this!

參考網址: https://medium.com/analytics-vidhya/kaggle-titanic-survival-prediction-top-3-ea6c8dcc9b6c