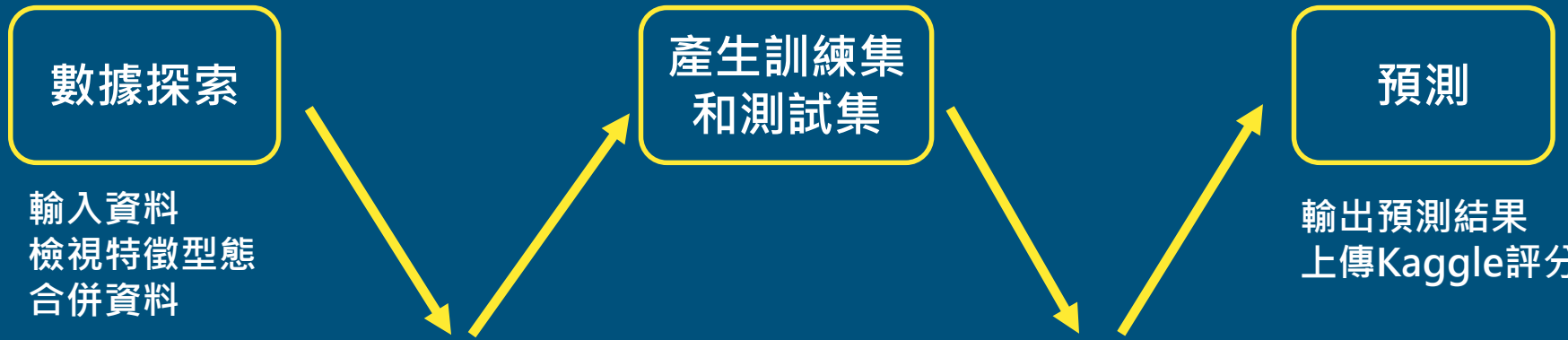
鐵達尼號生存預測 Kaggle競賽

報告人：407170070 陳煊凱

流程圖


數據探索
輸入資料
檢視特徵型態
合併資料
特徵工程
稱謂整理
填補遺漏值
調整數據型態
創造有利分析特徵
產生訓練集
和測試集
建立模型
調整參數
選擇特徵
餵入訓練資料
交叉驗證
預測
輸出預測結果
上傳Kaggle評分

數據探索

讀取資料

```
import pandas as pd
import numpy as np
df_train = pd.read_csv('titanic_train.csv').drop(['Unnamed: 0'],axis = 1)
df_test = pd.read_csv('titanic_test.csv').drop(['Unnamed: 0','Survived'],axis = 1)
df_train
```

檢視資料型態

```
df_train.shape , df_test.shape
```

```
((891, 12), (418, 11))
```

```
df_train["Age"].unique()
```

```
df_train.isnull( ).sum( )
```

合併train和test資料集

```
df_data = df_train.append( df_test )
df_data
```

特徵工程

整理稱謂

```
import re
regex = re.compile( ' ([A-Za-z]+)\.' )
df_data['Title'] = df_data.Name.map( lambda x:regex.search(x)[0] )
# Dropping the first and the last words
df_data['Title'] = df_data.Title.map( lambda x:x[1:][:-1] )
df_data['Title'].unique()
```

```
array(['Mr', 'Mrs', 'Miss', 'Master', 'Don', 'Rev', 'Dr', 'Mme', 'Ms',
       'Major', 'Lady', 'Sir', 'Mlle', 'Col', 'Capt', 'Countess',
       'Jonkheer', 'Dona'], dtype=object)
```

```
df_data['Title'].count()

df_data['Title'] = df_data.Title.replace( ['Don','Rev','Dr','Major','Lady','Sir','Col'
                                           ,'Capt','Countess','Jonkheer','Dona'], 'Rare' )
df_data['Title'] = df_data.Title.replace( ['Ms','Mlle'], 'Miss' )
df_data['Title'] = df_data.Title.replace( 'Mme', 'Mrs' )
df_data['Title'].unique()
```

```
array(['Mr', 'Mrs', 'Miss', 'Master', 'Rare'], dtype=object)
```

根據稱謂求年齡中位數及平均

```
Age_Mean = df_data[['Title','Age']].groupby( by=['Title'] ).mean()
Age_Median = df_data[['Title','Age']].groupby( by=['Title'] ).median()

Age_Mean.columns = ['Age Mean']
Age_Median.columns = ['Age Median']
Age_Mean.reset_index( inplace=True )
Age_Median.reset_index( inplace=True )

display( Age_Median )
display( Age_Mean )
```

	Title	Age Median
0	Master	4.0
1	Miss	22.0
2	Mr	29.0
3	Mrs	35.0
4	Rare	47.5

	Title	Age Mean
0	Master	5.482642
1	Miss	21.824366
2	Mr	32.252151
3	Mrs	36.918129
4	Rare	45.178571

填補年齡遺漏值

```
df_data.loc[(df_data.Age.isnull())&(df_data.Title=='Master'),'Age'] = Age_Mean.loc[Age_Mean.Title=='Master','Age Mean'][0]
df_data.loc[(df_data.Age.isnull())&(df_data.Title=='Miss'),'Age'] = Age_Mean.loc[Age_Mean.Title=='Miss','Age Mean'][1]
df_data.loc[(df_data.Age.isnull())&(df_data.Title=='Mr'),'Age'] = Age_Mean.loc[Age_Mean.Title=='Mr','Age Mean'][2]
df_data.loc[(df_data.Age.isnull())&(df_data.Title=='Mrs'),'Age'] = Age_Mean.loc[Age_Mean.Title=='Mrs','Age Mean'][3]
df_data.loc[(df_data.Age.isnull())&(df_data.Title=='Rare'),'Age'] = Age_Mean.loc[Age_Mean.Title=='Rare','Age Mean'][4]
df_data
```

根據Pclass求票價平均值

```
Pclass_Fare = df_data[['Pclass','Fare']].groupby(by=['Pclass']).mean()
Pclass_Fare
Pclass_Fare.columns = ['Fare Mean']
Pclass_Fare.reset_index( inplace = True)
Pclass_Fare
```

	Pclass	Fare Mean
0	1	88.809301
1	2	21.505098
2	3	13.340896

填補票價遺漏值

```
df_data.loc[ (df_data.Fare.isnull()) & (df_data.Pclass==3), 'Fare' ] = Pclass_Fare.loc[Pclass_Fare.Pclass==3,'Fare Mean'][2]
df_data.loc[ (df_data.Fare<=0) & (df_data.Pclass==1), 'Fare'] = Pclass_Fare.loc[Pclass_Fare.Pclass==1,'Fare Mean'][0]
df_data.loc[ (df_data.Fare<=0) & (df_data.Pclass==2), 'Fare'] = Pclass_Fare.loc[Pclass_Fare.Pclass==2,'Fare Mean'][1]
df_data.loc[ (df_data.Fare<=0) & (df_data.Pclass==3), 'Fare'] = Pclass_Fare.loc[Pclass_Fare.Pclass==3,'Fare Mean'][2]
df_data
```

調整數據型態 - 票價取Log

```
# 對 Fare 欄位取對數
df_data['LogFare'] = np.log1p( df_data.Fare )
# Label Encoding
Sex_mapping = { 'male':0, 'female':1 }
df_data[ 'Sex' ] = df_data.Sex.map( Sex_mapping )
```

創造有利分析特徵 - Sex_Pclass

```
import numpy as np
Survival_Rate = df_data[['Sex','Pclass','Survived']].groupby(by=['Sex','Pclass']).agg(np.mean)*100
Survival_Rate.columns = ['Survival Rate(%)']
Survival_Rate.reset_index()
```

	Sex	Pclass	Survival Rate(%)
0	female	1	96.808511
1	female	2	92.105263
2	female	3	50.000000
3	male	1	36.885246
4	male	2	15.740741
5	male	3	13.544669

將Sex_Pclass編碼

```
df_data.loc[ (df_data.Sex=='female') & (df_data.Pclass==1), 'Sex_Pclass' ] = 6
df_data.loc[ (df_data.Sex=='female') & (df_data.Pclass==2), 'Sex_Pclass' ] = 5
df_data.loc[ (df_data.Sex=='female') & (df_data.Pclass==3), 'Sex_Pclass' ] = 4
df_data.loc[ (df_data.Sex=='male') & (df_data.Pclass==1), 'Sex_Pclass' ] = 3
df_data.loc[ (df_data.Sex=='male') & (df_data.Pclass==2), 'Sex_Pclass' ] = 2
df_data.loc[ (df_data.Sex=='male') & (df_data.Pclass==3), 'Sex_Pclass' ] = 1
```

創造有利分析特徵 - Family_size

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

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

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

將Family_size和其他特徵做Dataframe

```
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	106	2	1.0
234	Cumings, Mr. John Bradley	PC 17599	71.2833	106	2	NaN
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	113803	53.1000	70	2	1.0
137	Futrelle, Mr. Jacques Heath	113803	53.1000	70	2	0.0
6	McCarthy, Mr. Timothy J	17463	51.8625	163	1	0.0
146	Hilliard, Mr. Herbert Henry	17463	51.8625	163	1	NaN
7	Palsson, Master. Gosta Leonard	349909	21.0750	-1	5	0.0
24	Palsson, Miss. Torborg Danira	349909	21.0750	-1	5	0.0
374	Palsson, Miss. Stina Viola	349909	21.0750	-1	5	0.0

有關聯的乘客 - Connected_survival

```
# 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
            elif (smin==0.0):
                df_data.loc[df_data['PassengerId'] == passID, 'Connected_Survival'] = 0
#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)
```

Connected_Survival	Survived
0.0	0.225
0.5	0.298
1.0	0.728

產生訓練測試集

X_train y_train test

```
# 產生訓練集和測試集
train = df_data[ pd.notnull(df_data.Survived) ]
test = df_data[ pd.isnull(df_data.Survived) ]

# 訓練集刪除 PassengerId 欄位；
# 測試集刪除 PassengerId 與 Survived 欄位
train.drop( ['PassengerId'], axis=1, inplace=True )
test.drop( ['Survived'], axis=1, inplace=True )

# 將測試集中的標籤欄位 Survived 單獨拆出
y_train = train.Survived
X_train = train.drop( ['Survived'], axis=1 )

X_train.shape,y_train.shape,test.shape
```

```
((891, 16), (891,), (418, 17))
```

預測

建立模型：隨機森林

```
# 隨機森林(Random Forest)
from sklearn.ensemble import RandomForestClassifier

Forest = RandomForestClassifier(n_estimators = 1000,
                                min_samples_split = 50,
                                min_samples_leaf = 1,
                                random_state = 1)
```

餵入模型

```
# 篩選部份特徵欄位餵入模型進行訓練
cols = ['Age','Sex','LogFare','Title','Pclass','Sex_Pclass','Connected_Survival','Family_size']
Forest.fit( X_train[cols], y_train )
y_pred = Forest.predict( X_train[cols])

print( f'Selected Features :\n   {cols}' )
print( 'train score',Forest.score( X_train[cols], y_train ) )
```

```
Selected Features :
   ['Age', 'Sex', 'LogFare', 'Title', 'Pclass', 'Sex_Pclass', 'Connected_Survival', 'Family_size']
train score 0.8496071829405163
```

K-fold交叉驗證

```
from sklearn.model_selection import cross_val_score, KFold
kfold =KFold(n_splits=5, shuffle=True)
Kf_cv_scores = cross_val_score(Forest,X_train[cols], y_train,cv=kfold)
print(Kf_cv_scores)
print('mean of K fold=',Kf_cv_scores.mean())
```

```
[0.83798883 0.84269663 0.85393258 0.84831461 0.84269663]
mean of K fold= 0.8451258552507689
```

預測結果輸出

```
Test_pred = Forest.predict(test[cols] )
print( Forest.predict_proba(test[cols]) )

submit = test[['PassengerId']]
submit ['Survived']=Test_pred
submit.to_csv('answer_1.csv')
submit
```

Name	Submitted	Wait time	Execution time	Score
answer_1.csv	just now	0 seconds	0 seconds	0.79425

Complete

建立模型：XGBoost

In [90]:

```
from xgboost import XGBClassifier
xgbc = XGBClassifier(n_estimators=5000
                     ,max_depth=50
                     ,objective = 'multi:softmax'
                     ,num_class = 10)
cols_2 = ['Sex','LogFare','Sex_Pclass','Family_size','Connected_Survival']
```

餵入模型

```
xgbc.fit(X_train[cols_2], y_train)
xgbc.score(X_train[cols_2],y_train)
```

```
0.9337822671156004
```

K-fold交叉驗證

```
from sklearn.model_selection import cross_val_score, KFold
kfold =KFold(n_splits=5, shuffle=True)
Kf_cv_scores = cross_val_score(xgbc,X_train[cols_2], y_train,cv=kfold)
print(Kf_cv_scores)
print('mean of K fold=',Kf_cv_scores.mean())
```

```
[0.81564246 0.84269663 0.84831461 0.83707865 0.79213483]
mean of K fold= 0.8271734354403364
```

預測結果輸出

```
Test_pred_2 =xgbc.predict( test[cols_2] )
print( xgbc.predict_proba(test[cols_2]) )
submit = test[['PassengerId']]
submit ['Survived']=Test_pred_2
submit.to_csv('answer_2.csv')
submit
```

Name	Submitted	Wait time	Execution time	Score
answer_2.csv	just now	0 seconds	0 seconds	0.78468

Complete

建立模型：logistic迴歸

```
from sklearn.linear_model import LogisticRegression
cols_3=['Age','Sex','LogFare','Title','Pclass','Sex_Pclass','Connected_Survival','Family_size']
Model = LogisticRegression(solver='lbfgs', multi_class='auto', max_iter=500,C=1.0)
Model.fit(X_train[cols_3], y_train)
```

餵入模型

```
Model.fit(X_train[cols_3], y_train)
```

```
Model.score(X_train[cols_3], y_train)
```

```
0.8316498316498316
```

K-fold交叉驗證

```
from sklearn.model_selection import cross_val_score, KFold
kfold =KFold(n_splits=5, shuffle=True)
Kf_cv_scores = cross_val_score(Model,X_train[cols_3], y_train,cv=kfold)
print(Kf_cv_scores)
print('mean of K fold=',Kf_cv_scores.mean())
```

```
[0.84357542 0.85393258 0.79775281 0.78651685 0.86516854]
mean of K fold= 0.8293892411022534
```

預測結果輸出

```
Test_pred_3  = Model.predict( test[cols_3] )
print( Model.predict_proba(test[cols_3]) )
submit = test[['PassengerId']]
submit ['Survived']=Test_pred_3
submit.to_csv('answer_3.csv')
submit
```

Name	Submitted	Wait time	Execution time	Score
answer_3.csv	just now	0 seconds	0 seconds	0.77990

Complete

心得

這次的報告讓我對機器學習這塊更有興趣，透過選用特徵和調整模型，上傳kaggle會得到不同的分數，讓我覺得非常具有挑戰性，為此我還創了三個kaggle帳號，雖然最後我得到的分數沒有很高，但這讓我認識到自己還有很多不足的地方，像是在特徵選用及調整上，我認為我還有很大的進步空間，只要特徵選得夠好，模型參數不用花太多的心力。

雖然這次報告做得並不好，但我寒假會繼續複習課程內容，以後kaggle有什麼初學競賽我還想試試看，希望我能繼續往這方面探索並前進。

未來方向

因為沒什麼學分的壓力，所以下學期我想看看有沒有我有興趣的課程，包括App開發或是AI學習的課程，對App有興趣是源自於奕兆老師的課，對AI學習有興趣是來自這次的報告，如果未來能在這兩個方面有所成就，或是能在畢業前找到這類的實習，可能就不會念研究所，因為覺得在沒有確定方向的情況下繼續進修非常不值得，費時又費力。

簡言之就是想往程式方面前進，如果能結合數學方面的知識，像是機器學習，更能凸顯我們念數學系的優勢。