

# 機器學習導論 - 期末報告

## Airbnb New User Bookings

巨資四 A 05170138 黃品嘉

巨資四 A 05170132 魏嗣宸

巨資四 A 05170120 王宇婕



## 目錄 Contents

---

一、	資料描述 Data Descriptions-----	3
二、	資料預處理 Data Preprocessing -----	4
三、	演算法測試 Test of Algorithm-----	12
	• xgboost	
	• logistic	
	• decision tree	
	• random forest	
四、	演算法成績比較 Algorithm Comparison-----	15
五、	結果 Conclusions-----	17
六、	參考資料 Reference-----	18

## 一、資料型態描述 Data Descriptions

- id: 使用者 id
- date\_account\_created: 帳號建立的日期
- timestamp\_first\_active: 使用者第一次使用的時間
- date\_first\_booking: 使用者第一次訂房的時間
- gender: 使用者性別
- age: 使用者年齡
- signup\_method: 註冊方式
- signup\_flow: 註冊頁面
- language: 是用語言
- affiliate\_channel: 付費管道
- affiliate\_provider: 付費管道名稱
- first\_affiliate\_tracked: 註冊前，接觸的付費管道
- signup\_app: 註冊 APP
- first\_device\_type: 第一次使用的設備類別
- first\_browser: 第一次使用的瀏覽器
- country\_destination: 訂房目的地，此項為本次的目標

## 二、資料預處理 Data Preprocessing

### 載入套件

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

### 讀取檔案

```
train = pd.read_csv(r"C:\Users\user\data\train_users_2.csv",dtype={'date_first_booking': str, 'date_account_created': str, 'timestamp_first_active':float})

test = pd.read_csv(r"C:\Users\user\data\test_users.csv",dtype={'date_first_booking': str, 'date_account_created': str, 'timestamp_first_active':float})

data = pd.concat([train , test] , axis = 0 , ignore_index=True)
```

設定 date\_first\_booking , date\_account\_created 為字串，

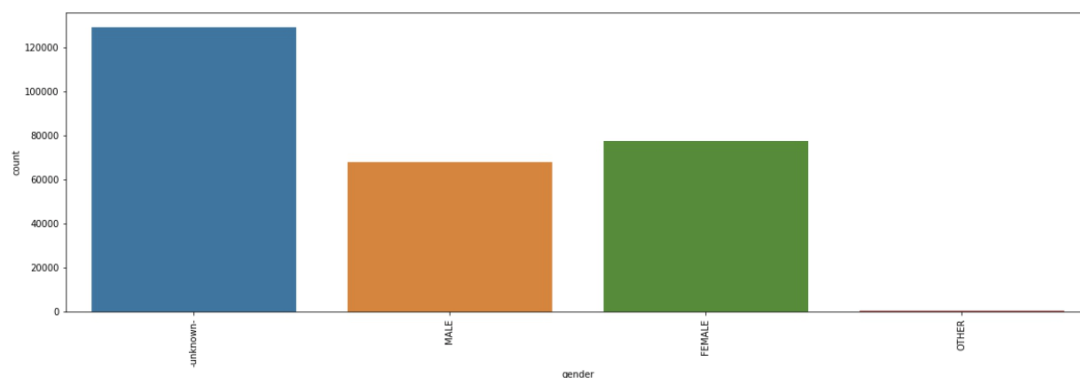
timestamp\_first\_active 為浮點數，合併成 data 檔。

### 處理 Gender 資料

```
plt.figure(figsize=(20,6))
sns.countplot(data.gender)
plt.xticks(rotation=90)
```

### 觀察發現-unknown-比例很高

(array([0, 1, 2, 3]), <a list of 4 Text xticklabel objects>)



```
data.gender.replace('-unknown-', np.nan, inplace=True)
print("New null value % in gender is: " + "{0:.2%}".format(sum(data.gender.isnull())/
data.shape[0]))
```

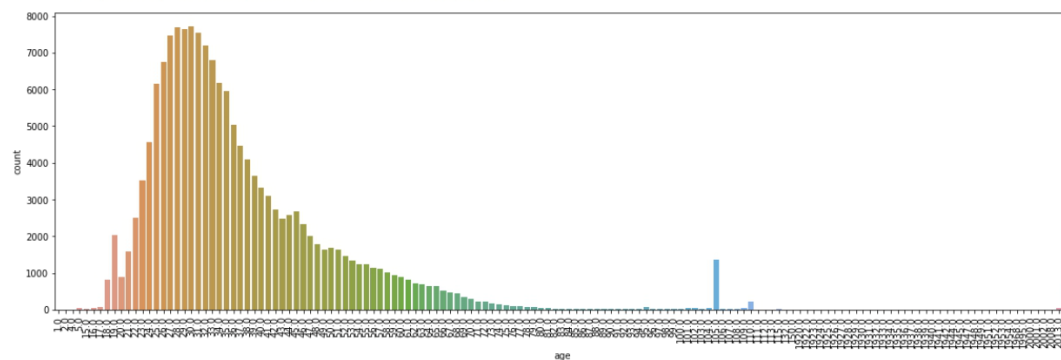
將-unknown-替換成空值，並檢視空值的比率為 46.99%

## 處理 age 資料

```
plt.figure(figsize=(20,6))
sns.countplot(data.age)
plt.xticks(rotation=90)
```

觀察 age 分佈，發現年齡範圍有超出合理範圍的值 Ex: 2014

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,
        13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
        26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
        39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
        52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64,
        65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
        78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90,
        91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103,
        104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
        117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
        130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
        143, 144]), <a list of 145 Text xticklabel objects>)
```



```
data.loc[data.age < 18, 'age'] = np.nan
data.loc[data.age > 80, 'age'] = np.nan
data.age = data.age.replace("NaN", np.nan)
print("Now the % of null values in age is: " + "{0:.2%}".format(sum(data.age.isnull())
/data.shape[0]))
```

將 18 歲以下、80 歲以上視為錯誤資料→替換成空值，並檢視空值的比率為

43.64%

## 處理 signup\_flow 資料

```
data['signup_flow'] = data['signup_flow'].astype('object')
```

將 signup\_flow 轉成類別型態 ( 後會將資料轉成虛擬列 get\_dummies , 故轉成類別型態 )

## 處理 affiliate\_channel 資料

```
data.affiliate_channel.value_counts()
```

direct	181571
sem-brand	36439
sem-non-brand	20075
seo	14362
other	9547
api	8167
content	4118
remarketing	1268

```
data.affiliate_channel.replace('api', 'other', inplace=True)  
data.affiliate_channel.replace('content', 'other', inplace=True)  
data.affiliate_channel.replace('remarketing', 'other', inplace=True)
```

將數量較低的類別都歸類到 other

## 處理 affiliate\_provider 資料

```
data.affiliate_provider.value_counts()
```

direct	181270
google	65956
other	13036
facebook	3996
bing	3719
craigslist	3475
padmapper	836
vast	830
yahoo	653
facebook-open-graph	566
gsp	455
meetup	358
email-marketing	270
naver	66
baidu	32
yandex	18
wayn	8
daum	3

```
data.affiliate_provider.replace('daum', 'other', inplace=True)
data.affiliate_provider.replace('wayn', 'other', inplace=True)
data.affiliate_provider.replace('yandex', 'other', inplace=True)
data.affiliate_provider.replace('baidu', 'other', inplace=True)
data.affiliate_provider.replace('naver', 'other', inplace=True)
data.affiliate_provider.replace('email-marketing', 'other', inplace=True)
data.affiliate_provider.replace('meetup', 'other', inplace=True)
.....
```

將數量較低的類別都歸類 other

## 處理 first\_browser 資料

```
data.first_browser.replace('IE Mobile', 'other', inplace=True)
data.first_browser.replace('AOL Explorer', 'other', inplace=True)
data.first_browser.replace('Android Browser', 'other', inplace=True)
data.first_browser.replace('Chrome Mobile', 'other', inplace=True)
data.first_browser.replace('Chromium', 'other', inplace=True)
data.first_browser.replace('BlackBerry Browser', 'other', inplace=True)
data.first_browser.replace('IE Mobile ', 'other', inplace=True)
data.first_browser.replace('Silk', 'other', inplace=True)
.....
```

將數量較低的類別都歸類 other

- affiliate\_channel、affiliate\_provider、first\_browser 將部分類別合併到

other 是為了避免 get dommies 完後變成太多欄位，以致最後跑演算法負

荷過重，但也不能取代過多，以免結果產生偏差。如下圖所示：

Your most recent submission				
Name	Submitted	Wait time	Execution time	Score
submission.csv	just now	0 seconds	6 seconds	0.85772
Complete				
<a href="#">Jump to your position on the leaderboard</a> ▼				



Your most recent submission				
Name	Submitted	Wait time	Execution time	Score
submission.csv	2 days ago	0 seconds	6 seconds	0.85686
Complete				
<a href="#">Jump to your position on the leaderboard</a> ▼				



## one-hot-encoding-1

```
columns = ['affiliate_channel', 'affiliate_provider',
           'first_affiliate_tracked',
           'first_browser', 'first_device_type',
           'signup_app', 'signup_method']
for c in columns:
    data_ohe = pd.get_dummies(data[c], prefix=c, dummy_na=True)
    data.drop([c], axis = 1, inplace = True)
    data = pd.concat((data, data_ohe), axis = 1)
```

將部分類別資料轉成虛擬列

## age 空值處理

```
from sklearn.linear_model import LogisticRegression
def set_missing_ages(df):
    columns = df[['age', 'affiliate_channel_direct',
                  'affiliate_channel_other', 'affiliate_channel_sem-brand',
                  'affiliate_channel_sem-non-brand', 'affiliate_channel_seo',
                  'affiliate_channel_nan', 'affiliate_provider_bing',
                  'affiliate_provider_craigslist',
                  .....]]
    # 分成已知年齡和未知年齡兩部分
    known_age = columns[columns.age.notnull()].as_matrix()
    unknown_age = columns[columns.age.isnull()].as_matrix()

    y = known_age[:, 0]
    X = known_age[:, 1:]

    lr=LogisticRegression()
    lr.fit(X, y)

    predictedages = lr.predict(unknown_age[:, 1:])

    df.loc[ (df.age.isnull()), 'age' ] = predictedages

    return df, lr
```

以邏輯式迴歸訓練已知年齡資料預測 age 的空值

## gender 空值處理

```
data.gender.replace('MALE', 0, inplace=True)
data.gender.replace('FEMALE', 1, inplace=True)
data.gender.replace('OTHER', 2, inplace=True)
```

先將文字轉成數值，方便後面跑演算法預測

```
def set_missing_genders(df):
    columns = df[['gender', 'affiliate_channel_direct',
                  'affiliate_channel_other', 'affiliate_channel_sem-brand',
                  'affiliate_channel_sem-non-brand',
                  .....]]

    known_gender = columns[columns.gender.notnull()].as_matrix()
    unknown_gender = columns[columns.gender.isnull()].as_matrix()

    y = known_gender[:, 0]
    X = known_gender[:, 1:]

    lr=LogisticRegression()
    lr.fit(X, y)

    predictedgenders = lr.predict(unknown_gender[:, 1:])
    df.loc[ (df.gender.isnull()), 'gender' ] = predictedgenders

    return df, lr
```

以邏輯式迴歸訓練已知性別資料預測 gender 的空值

## age 資料分類

```
bins=[17,25,35,45,55,65,81]
labels=['18-25', '26-35', '36-45', '46-55', '56-65', '66-80',]
data['age']=pd.cut(data.age,bins,labels=labels)
```

將 age 分成 6 個階段 ( 數值轉類別 )

## timestamp\_first\_active 資料分類

```
bins=[20081200000000,20090131246060,20090531246060,20090831246060
      ,20091130246060,20100131246060,20100531246060,20100831246060
      ,20101130246060,20110131246060,20110531246060,20110831246060
      ,20111130246060,20120131246060,20120531246060,20120831246060
      ,20121130246060,20130131246060,20130531246060,20130831246060
      ,20131130246060,20140131246060,20140531246060,20140831246060
      ,20141130246060,20150131246060]
labels=['0812-0902','0903-0905','0906-0908','0909-0911','0912-1002'
        , '1003-1005','1006-1008','1009-1011','1012-1102'
        , '1103-1105','1106-1108','1109-1111','1112-1202'
        , '1203-1205','1206-1208','1209-1211','1212-1302'
        , '1303-1305','1306-1308','1309-1311','1312-1402'
        , '1403-1405','1406-1408','1409-1411','1412-1502']

data['timestamp_first_active']=pd.cut(data.timestamp_first_active,
                                     bins,labels=labels)
```

將 timestamp\_first\_active 以年度及季節分成 25 個階段 ( 數值轉類別 , Ex:

0812-0902→冬 )

## 刪除欄位

```
data.drop(labels=['date_first_booking'],axis='columns',inplace=True)
data.drop(labels=['id'],axis='columns',inplace=True)
```

id 非特徵欄位 ; data\_first\_booking 在 test 檔中全為空值 , 因此這刪掉 2 欄位

## one-hot-encoding-2

```
columns = ['age', 'date_account_created', 'gender',
           'language', 'signup_flow', 'timestamp_first_active']
for c in columns:
    data_ohe = pd.get_dummies(data[c],prefix=c, dummy_na=True)
    data.drop([c], axis = 1, inplace = True)
    data = pd.concat((data, data_ohe), axis = 1)
```

## 目標欄位轉 label-encoding

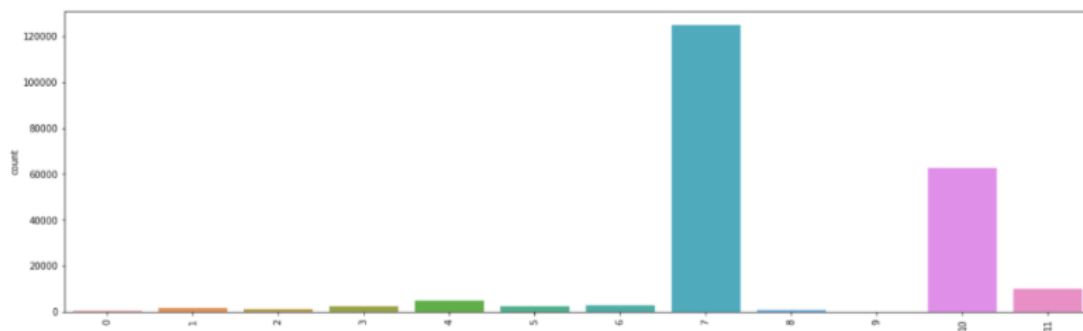
```
y = train['country_destination'].values

from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
y = labelencoder.fit_transform(y)
```

將要預測的國家轉成代碼

```
plt.figure(figsize=(20,6))
sns.countplot(y)
plt.xticks(rotation=90)
```

(array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]),  
<a list of 12 Text xticklabel objects>)



## 三、 演算法測試 Test of Algorithm

切割 train · test 特徵欄位

```
X_train = data.values[:213451]
X_test = data.values[213451:]
```

test id 抓取

```
test_ids = test['id']
```

## 1. xgboost

```
import xgboost as xgb
from xgboost.sklearn import XGBClassifier

xgb = XGBClassifier(max_depth=6, learning_rate=0.3, n_estimators=22,
                    objective='multi:softprob', subsample=0.6, colsample_bytree=0.6, seed=0)
xgb.fit(X_train, y)
y_pred = xgb.predict_proba(X_test)
```

預測結果示意如下：

```
y_pred
array([[0.00309393, 0.00504772, 0.00429806, ..., 0.00238365, 0.21934034, 0.03065487], ...,
       [0.00380462, 0.00760705, 0.0051426, ..., 0.00231592, 0.37696147, 0.04176515]], dtype=float32)
```

## submission 存檔

```
ids = [] #list of ids
cts = [] #list of countries
for i in range(len(test_ids)):
    idx = test_ids[i]
    ids += [idx] * 5
    cts += labelencoder.inverse_transform(np.argsort(y_pred[i])[:, :-1])[:5].tolist()

sub = pd.DataFrame(np.column_stack((ids, cts)), columns=['id', 'country'])
sub.to_csv('submission.csv', index=False)
```

kaggle 成績：0.85686

Name	Submitted	Wait time	Execution time	Score
xgboost.csv	just now	0 seconds	6 seconds	0.85686
Complete				
<a href="#">Jump to your position on the leaderboard</a> ▼				

## 2. logistic

```
lr=LogisticRegression()
lr.fit(X_train, y)
y_pred = lr.predict_proba(X_test)
```

kaggle 成績：0.85467

Name	Submitted	Wait time	Execution time	Score
logistic.csv	just now	0 seconds	6 seconds	0.85467
Complete				
<a href="#">Jump to your position on the leaderboard</a> ▼				

### 3. decision tree

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(criterion = 'gini',
                             random_state = 1,
                             max_depth = 5,)

tree.fit(X_train,y)
y_pred_tree = tree.predict_proba(X_test)
```

kaggle 成績 : 0.85527

Name	Submitted	Wait time	Execution time	Score
tree.csv	just now	0 seconds	5 seconds	0.85527
Complete				
<a href="#">Jump to your position on the leaderboard</a> ▼				

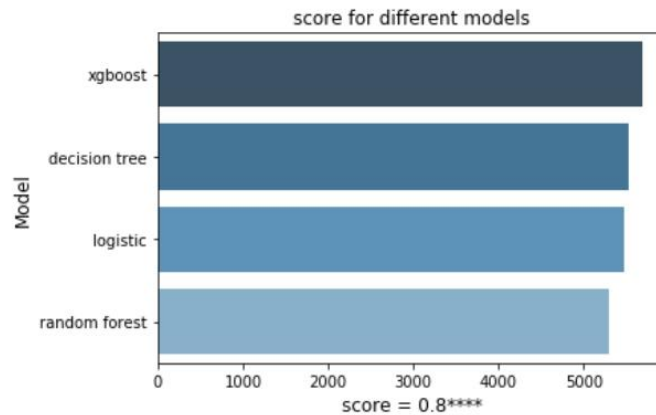
### 4. random forest

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=300,criterion='gini')
rf.fit(X_train,y)
y_pred = rf.predict_proba(X_test)
```

kaggle 成績 : 0.85299

Name	Submitted	Wait time	Execution time	Score
submission10.csv	just now	0 seconds	6 seconds	0.85299
Complete				
<a href="#">Jump to your position on the leaderboard</a> ▼				

#### 四、演算法成績比較：



- 成績最好的為 **xgboost**

Boosting Tree 用 tree 來降低殘差，然後 xgboost 又引入二階導數（二階泰勒展開式）進行求解，引入節點數和參數的 L2 正則性來評估模型的複雜性，並構造 xgboost 的預測和目標函數。

- 優點：

1. 將樹形模型的複雜性（以正則項表示）添加到優化目標。
2. 在公式推導中使用二階導數，並使用二階泰勒展開式。
3. 實現了用於分割點查找的近似算法。
4. 利用特徵稀疏性。
5. 預先對數據進行排序並以塊形式存儲，這有利於並行計算。
6. 基於分佈式通訊框架 rabbit，它可以在 MPI 和 yarn 上運行。
7. 針對架構優化了實現，針對緩存和內存優化了性能。

- **xgboost 參數介紹**

1. **learning\_rate=0.3**：用於更新葉子節點權重時，乘以該係數，避免步長過大，引數值越大，越可能無法收斂，把學習率 **eta** 設定的小一些，小學習率可以使得後面的學習更加仔細。
2. **max\_depth =6**：每顆樹的最大深度，樹高越深，越容易過擬合。
3. **subsample=0.6**：樣本隨機取樣，較低的值使得演算法更加保守，防止過擬合，但是太小的值也會造成欠擬合。
4. **colsample\_bytree =0.6**：列取樣，對每棵樹的生成用的特徵進行列取樣。
5. **objective='multi:softprob'**：可以將該向量 reshape 成 `ndata` 行 `nclass` 列的矩陣。每行數據表示樣本所屬於每個類別的機率。
6. **seed=0**：隨機數種子設為 0。
7. **n\_estimators=40**：生成的最大樹數目



## 五、 結果 Conclusions

- xgboost 參數調整

### 1. max\_depth

max\_depth 4 = 0.85592

max\_depth 6 = 0.85686

max\_depth 8 = 0.85784

max\_depth 10 = 0.85788 → 雖然分數較高，但不明顯，而且跑演算法過久，模型太複雜，不予取用。

### 2. learning\_rate

learning\_rate 0.4 = 0.85775

learning\_rate 0.3 = 0.85784

learning\_rate 0.2 = 0.85763

### 3. n\_estimators

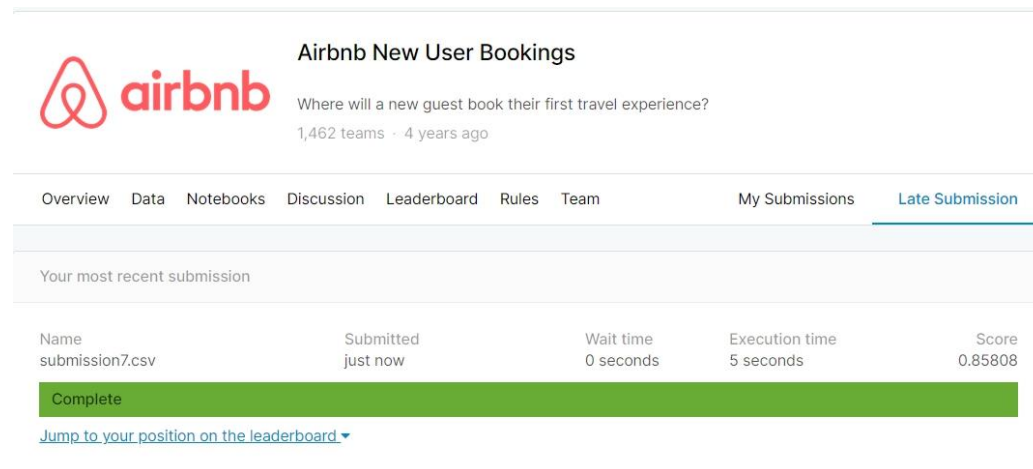
n\_estimators 50 = 0.85759

n\_estimators 40 = 0.85808

n\_estimators 22 = 0.85784

n\_estimators 12 = 0.85750

調整完參數後，成績從 0.85686 提升至 0.85808。



Airbnb New User Bookings

Where will a new guest book their first travel experience?

1,462 teams · 4 years ago

Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions **Late Submission**

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission7.csv	just now	0 seconds	5 seconds	0.85808

Complete

[Jump to your position on the leaderboard](#)

➤ 最終以 XGBoost 演算法在以下參數值訓練出成績最好的模型

```
xgb =XGBClassifier( max_depth = 8 ,  
                    learning_rate = 0.3 ,  
                    n_estimators = 40 ,  
                    objective = 'multi:softprob' ,  
                    subsample = 0.6 ,  
                    colsample_bytree = 0.6 ,  
                    seed = 0 )
```

## 六、參考資料 Reference

1. <https://medium.com/@PatHuang/%E5%88%9D%E5%AD%B8python%E6%89%8B%E8%A8%98-3-%E8%B3%87%E6%96%99%E5%89%8D%E8%99%95%E7%90%86-label-encoding-one-hot-encoding-85c983d63f87>
2. <https://blog.csdn.net/Datawhale/article/details/80847662>
3. <https://zhuanlan.zhihu.com/p/28672955>