巨資四 B 機器學習導論 期末報告 ——

WSDM - KKBox's Music Recommendation Challenge

KKBOX 歌曲推薦系統

一、 競賽資訊介紹

- 競賽題目：
  WSDM - KKBox's Music Recommendation Challenge（KKBOX 歌曲推薦系統）
- 目標：
  預測用戶在一段時間內觸發第一個可觀察到的收聽事件後重複播放歌曲的機會。
- 資料集：
  1. train.csv：此競賽資料提供的訓練集。
  2. test.csv：此競賽資料提供的測試集。
  3. songs.csv：內含與歌曲相關的資料，例如：歌曲長度、風格、作詞、作曲者、歌詞、語言。
  4. members.csv：內含與使用者相關資訊，例如：使用者的城市、年齡、性別、註冊方法及日期、到期日。
  5. song_extra_info.csv：此競賽資料提供的屬於歌曲的額外資訊。例如：曲名、ISRC（國際標準錄音／錄影資料代碼）。
- 評估方式：
  利用 ROC 曲線（受試者工作特徵曲線）評估提交的答案內容中預測概率和觀測目標之間的準確率，ROC 為一個比 Accuracy、Precision、Recall、F1-Score 更加全面評估效能的指標。

二、 建模前預先準備工作（資料前處理）

（一） 載入數據集並觀察各數據集

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
member = pd.read_csv('members.csv',parse_dates = ['registration_init_time','expiration_date'])
songs = pd.read_csv('songs.csv')
extra_song = pd.read_csv('song_extra_info.csv')
```

executed in 23.1s, finished 18:42:20 2021-01-19

```
print('train:',train.shape)
print('test:',test.shape)
print('member:',member.shape)
print('songs:',songs.shape)
print('extra_song', extra_song.shape)
```

executed in 19ms, finished 17:04:48 2021-01-18

```
train: (7377418, 6)
test: (2556790, 6)
member: (34403, 7)
songs: (2296320, 7)
extra_song (2295971, 3)
```

從上圖可發現各別資料集的概況。

1. train 資料集：有 7,377,418 筆資料、6 個欄位的資料集。

```
1 train.head()
```

executed in 10ms, finished 17:01:12 2021-01-18

	msno	song_id	source_system_tab	source_screen_name	source_type	target
0	FGtllVqz18RPiwJj/edr2gV78zirAiY/9SmYvia+kCg=	BBzumQNXUHKdEBOB7mAJuzok+IJA1c2Ryg/yzTF6tik=	explore	Explore	online-playlist	1
1	Xumu+NIjS6QYVxDS4/t3SawvJ7viT9hPKXmf0RtLNx8=	bhp/MpSNoqoxOIB+/l8WPqu6jldth4DIpCm3ayXnJqM=	my library	Local playlist more	local-playlist	1
2	Xumu+NIjS6QYVxDS4/t3SawvJ7viT9hPKXmf0RtLNx8=	JNWfrrC7zNN7BdMpsISKa4Mw+xVJYNnxXh3/Epw7QgY=	my library	Local playlist more	local-playlist	1
3	Xumu+NIjS6QYVxDS4/t3SawvJ7viT9hPKXmf0RtLNx8=	2A87tzfnJTSWqD7gIZHisolhe4DMdzkbd6LzO1KHjNs=	my library	Local playlist more	local-playlist	1
4	FGtllVqz18RPiwJj/edr2gV78zirAiY/9SmYvia+kCg=	3qm6XTZ6MOCU11x8FIVbAGH5l5uMkT3/ZalWG1oo2Gc=	explore	Explore	online-playlist	1

2. test 資料集：一個有 2,556,790 筆資料、6 個欄位的資料集。

```
1 test.head()
```

executed in 21ms, finished 16:57:55 2021-01-18

	id	msno	song_id	source_system_tab	source_screen_name	source_type
0	0	V8ruy7SGk7tDm3zA51DPpn6qutt+vmKMBKa21dp54uM=	WmHKgKMlp1IQMecNdNvDMkvlycZYHnFwDT72I5sIssc=	my library	Local playlist more	local-library
1	1	V8ruy7SGk7tDm3zA51DPpn6qutt+vmKMBKa21dp54uM=	y/rsZ9DC7FwK5F2PK2D5mj+aOBUJAjuu3dZ14NgE0vM=	my library	Local playlist more	local-library
2	2	/uQAlrAkaczV+nWCd2sPF2ekvXPRipV7q0l+gbLuxjw=	8eZLFOdGVdXBSqoAv5nsLigeH2BvKXzTQYtUM53I0k4=	discover	NaN	song-based-playlist
3	3	1a6oo/iXKatxQx4eS9zTVD+KlSVaAFbTIqVvwLC1Y0k=	ztCf8thYsS4YN3GcIL/bvoxLm/T5mYBVKOO4C9NiVfQ=	radio	Radio	radio
4	4	1a6oo/iXKatxQx4eS9zTVD+KlSVaAFbTIqVvwLC1Y0k=	MKVMpsIKcQhMaFEgcEQhEfi5+RZhMYlU3eRDpySrH8Y=	radio	Radio	radio

3. members 資料集：有 34,403 筆資料、7 個欄位的資料集。

```
1 member.head()
```

executed in 15ms, finished 16:58:09 2021-01-18

	msno	city	bd	gender	registered_via	registration_init_time	expiration_date
0	XQxgAYj3klVKjR3oxPPXYYFp4soD4TuBghkhMTD4oTw=	1	0	NaN	7	2011-08-20	2017-09-20
1	UizsfmJb9mV54qE9hCYyU07Va97c0lCRLEQX3ae+ztM=	1	0	NaN	7	2015-06-28	2017-06-22
2	D8nEhsIOBSoE6VthTaqDX8U6lqjJ7dLdr72mOyLya2A=	1	0	NaN	4	2016-04-11	2017-07-12
3	mCuD+tZ1hERA/o5GPqk38e041J8ZsBaLcu7nGoIIvhI=	1	0	NaN	9	2015-09-06	2015-09-07
4	q4HRBfVSssAFS9iRfxWrohxuk9kCYMKjHOEagUMV6rQ=	1	0	NaN	4	2017-01-26	2017-06-13

處理 members 的資料集中，載入資料時隨即利用 parse_dates 將初始註冊日期、到期日期從 int 轉變為日期格式，例如：第一筆資料中的 registration_init_time 在原先的 data 中為 20110820，故轉變後為 2011-08-20，而 expiration_date 亦然如此，將原先的 20170920 轉變成 2017-09-20。

4. songs 資料集：有 2,296,320 筆資料、7 個欄位的資料集。

```
1 songs.head()
```

executed in 16ms, finished 16:58:24 2021-01-18

	song_id	song_length	genre_ids	artist_name	composer	lyricist	language
0	CXoTN1eb7AI+DntdU1vbcwGRV4SCIDxZu+YD8JP8r4E=	247640	465	張信哲 (Jeff Chang)	董貞	何啟弘	3.0
1	o0kFgae9QtnYgRkVPqLJwa05zIhRlUjfF7O1tDw0ZDU=	197328	444	BLACKPINK	TEDDY| FUTURE BOUNCE| Bekuh BOOM	TEDDY	31.0
2	DwVvVurfpuz+XPuFvucclVQEyPqcpUkHR0ne1RQzPs0=	231781	465	SUPER JUNIOR	NaN	NaN	31.0
3	dKMBWoZyScdxSkihKG+Vf47nc18N9q4m58+b4e7dSSE=	273554	465	S.H.E	湯小康	徐世珍	3.0
4	W3bqWd3T+VeHFzHAUfARgW9AvVRaF4N5Yzm4Mr6Eo/o=	140329	726	貴族精選	Traditional	Traditional	52.0

5. extra_song 資料集：有 2,295,971 筆資料、 3 個欄位的資料集。

```
1 extra_song.head()
```

executed in 14ms, finished 16:59:09 2021-01-18

	song_id	name	isrc
0	LP7pLJoJFBvyuUwvu+oLzjT+bI+UeBPURCecJsX1jjs=	我們	TWUM71200043
1	ClazTFnk6r0Bnuie44bocdNMM3rdlrq0bCGAsGUWcHE=	Let Me Love You	QMZSY1600015
2	u2ja/bZE3zhCGxvbbOB3zOoUjx27u40cf5g09UXMoKQ=	原諒我	TWA530887303
3	92Fqsy0+p6+RHe2EoLKjHahORHR1Kq1TBJoCIW9v+Ts=	Classic	USSM11301446
4	0QFmz/+rJy1Q56C1DuYqT9hKKqi5TUqx0sN0IwvoHrw=	愛投羅網	TWA471306001

（二） 觀察是否有遺漏值

透過 msno、song_id 可發現 train、test 雖然其資料筆數達七百多萬及二百萬多筆，但其中唯一（即不重複）的使用者數量（30,755、25,131）及歌曲數量（359,966、224,753）皆小於前述觀察到的 members 資料集（34,403）中的資料筆數和 songs 資料集（2,296,320）中的資料筆數。故大膽假設所有的資料皆能透過 msno 及 son_id 尋找到對應的資訊。

```
1 print('train中唯一的msno(user_id):',len(train.msno.unique()))
2 print('train中唯一的song_id:',len(train.song_id.unique()))
3 print('test中唯一的msno(user_id):',len(test.msno.unique()))
4 print('test中唯一的song_id:',len(test.song_id.unique()))
```

executed in 3.97s, finished 17:05:00 2021-01-18

```
train中唯一的msno(user_id): 30755
train中唯一的song_id: 359966
test中唯一的msno(user_id): 25131
test中唯一的song_id: 224753
```

接著即將 trrain 和 test 兩個資料集皆以 how=left 方式合併左側的 DataFrame，故以 msno 合併 member 資料集，再以 song_id 合併 songs 資料集，合併後的資料集有 7,377,418 筆資料、18 個欄位的資料集 df_train。

（三） 資料清理

1. 遺漏值處理及補齊

先檢查遺漏值的數量及其遺漏佔總體的比率。從下圖中可發現有許多欄位的資料遺失率相當高，甚至 lyricist 及 gender 的遺失率更高達四成，故需要對遺失值做一些處理，填補或是刪除等動作，以利後續的資料分析。

	Quantity	Percent
lyricist	3178798	0.430882
gender	2961479	0.401425
composer	1675706	0.227140
source_screen_name	414804	0.056226
genre_ids	118455	0.016056
source_system_tab	24849	0.003368
source_type	21539	0.002920
language	150	0.000020
song_length	114	0.000015
artist_name	114	0.000015
registered_via	0	0.000000
registration_init_time	0	0.000000
bd	0	0.000000
city	0	0.000000
target	0	0.000000
expiration_date	0	0.000000
song_id	0	0.000000
msno	0	0.000000

接著即為填補遺漏值，由於 object 屬性是 pandas 表示 string 的資料格式，故當資料為 object 時，將遺漏值補上「unknown」，其餘的數值型態的缺漏值則是以 0 補齊。

```
#填補遺漏值
#object是pandas表示string的資料格式，當資料為object時，將遺漏值補上「unknown」，其他數值型態的闕漏值則補上「0」。
for i in df_train.select_dtypes(include = ['object']).columns:
    df_train[i][df_train[i].isnull()] = 'unknown'
df_train = df_train.fillna(value = 0)
```

2. 日期拆分（Y、M、D），刪除原日期並新增拆分後的六個日期。

```
# registration_init_time
df_train['registration_init_time_year'] = df_train['registration_init_time'].dt.year
df_train['registration_init_time_month'] = df_train['registration_init_time'].dt.month
df_train['registration_init_time_day'] = df_train['registration_init_time'].dt.day
df_train = df_train.drop(['registration_init_time'], axis = 1)

# expiration_date
df_train['expiration_date_year'] = df_train['expiration_date'].dt.year
df_train['expiration_date_month'] = df_train['expiration_date'].dt.month
df_train['expiration_date_day'] = df_train['expiration_date'].dt.day
df_train = df_train.drop(['expiration_date'], axis = 1)
```

把 registration_init_time、expiration_date 之日期由一組日期拆分成年、月、日三個欄位，並刪除原本欄位，故拆分後的 df_train 欄位共有 18-2+6 = 22 個欄位。會想要拆分是因為若是要做後續分析的話，可以觀察註冊的年、月、日何者最密集，亦可觀察使用者大多都是何時到期，以利後續推薦系統的分析。

三、 EDA（Exploratory Data Analysis）探索式資料分析

EDA 是運用基本敘述性統計、統計繪圖、視覺化等工具快速簡易的方式，從各種面向先了解資料概況，以對資料有初步認識，以利後續對資料進行複雜或嚴謹的分析能更精確的瞄準目標。而其主要能給予大大地幫助以認識資料中三個部分，分別是下列所述：

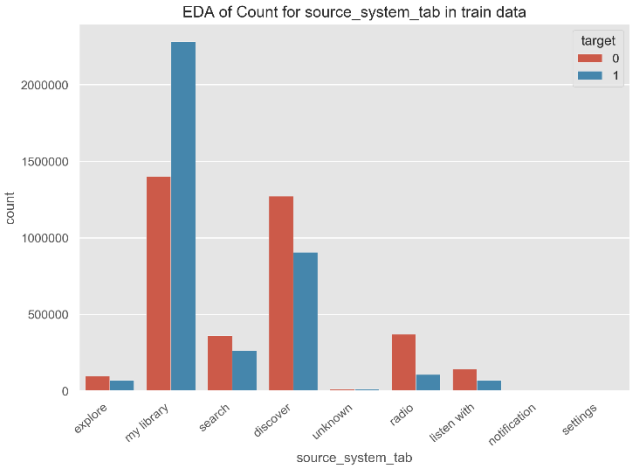
1. 瞭解資料，獲取資料的資訊、結構和特點。
2. 檢查有無離群值或異常值，看資料是否有誤。
3. 分析各變數之間的關聯性，以找出較重要的變數。
4. 檢查資料是否符合分析前的假設、在模型建立前先發現潛在的錯誤，並進一步調整整體的分析方向。

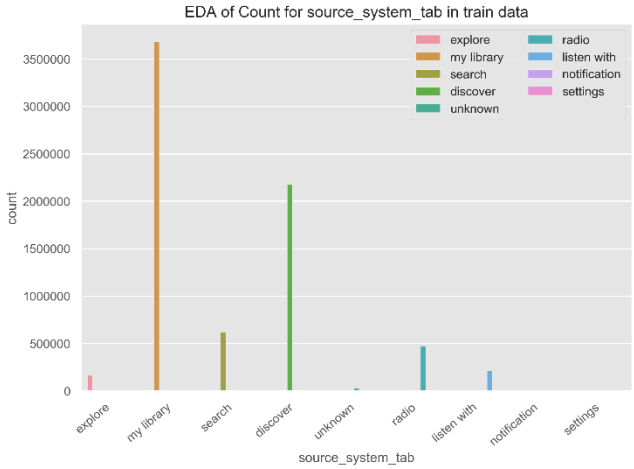
資料來源：淺談何謂探索式資料分析

https://ithelp.ithome.com.tw/articles/10213384

（一） Train.csv 中的 feature

1. source_system_tab





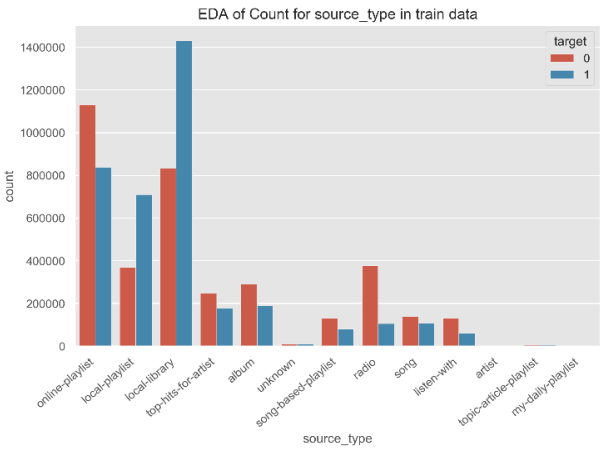
從上方右圖中可發現在 source_system_tab 數量最多的是 my library，次之為 discover，第三則為 radio，大多是跟用戶日常最常聽的模式有關，除了自身的歌單收藏外，尚且利用尋找功能搜尋想要聽的歌曲，再來就是電臺，可能最近很流行的 Podcast（播客）相關。

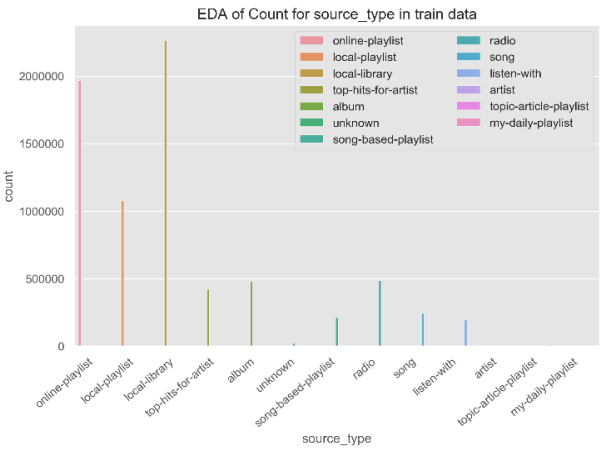
再把 source_system_tab 與 target 交叉比對並視覺化後發現，my library 和 discover 的 target = 1 的比例是較高的，且其二者的 target = 0 亦為此屬性中相對較高的，其他屬性的資料筆數都不多。

2. source_type

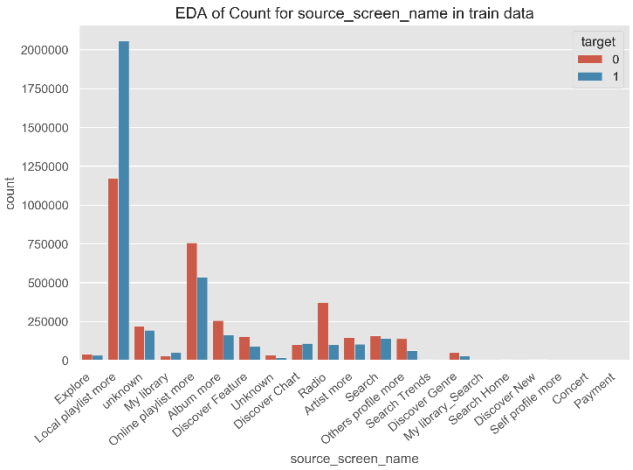
從下方右圖中可發現在 source_type 數量最多的是 local-library，次之為 online-playlist，第三則為 local-playlist，大多是跟 local 相關，推測其為 source_type 屬性中的主要來源。

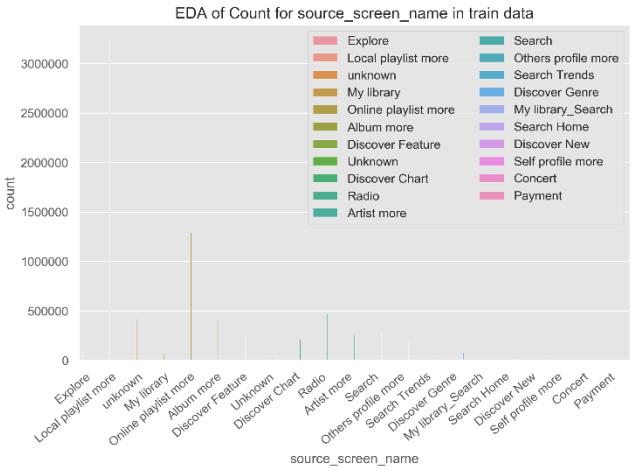
再把 source_type 與 target 交叉比對並視覺化後發現，local-library、online-playlist、local-playlist 的 target = 1 的比例是較高的，且前二者的 target = 0 亦為此屬性中相對較高的，其他屬性的資料筆數都相較不多。





3. source_screen_name





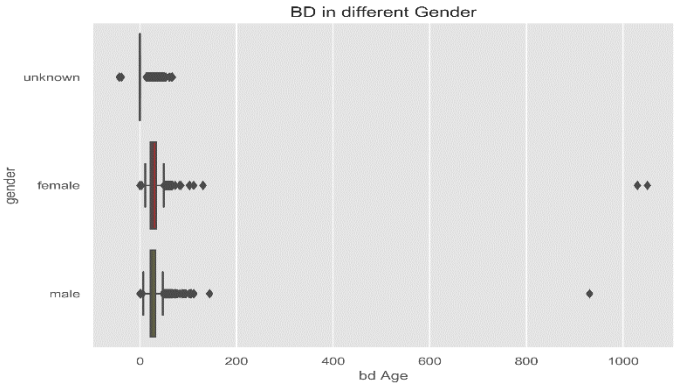
從上方右圖中可發現在 source_screen_name 數量最多的是 Local playlist more，次之為 Online Playlist more，第三則為 Radio，理由大致上與 source_system_tab 相同，主要猜測跟用戶日常最常聽的模式有較大的關連性。

再把 source_screen_name 與 target 交叉比對並視覺化後發現，Local playlist more 的 target = 1 高居第一，將其他屬性遠遠拋在後面，且其 target = 0 的比例亦是較高的，兩者較平均的尚且還有 Online Playlist more，其他屬性的資料筆數都不多。

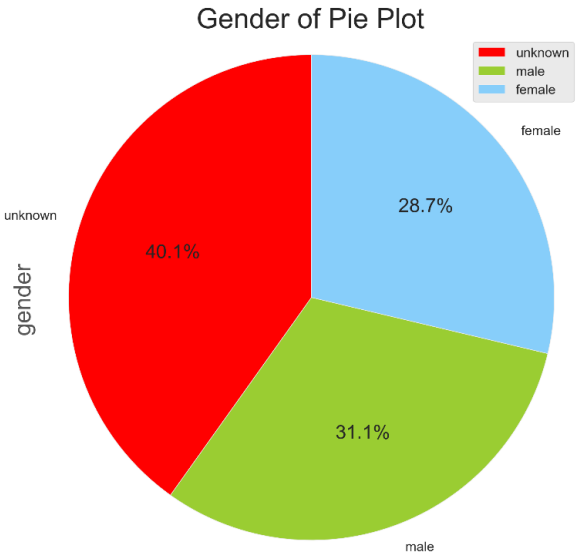
（二） Members.csv 中的 feature

1. bd、gender

從右側的盒狀圖可以發現 male、female，甚至是遺漏值 unknown，其對照的 bd（年齡）皆有異常值，且使用者的平均齡大多落在約 50 歲以內，推測以年輕族群為大宗。

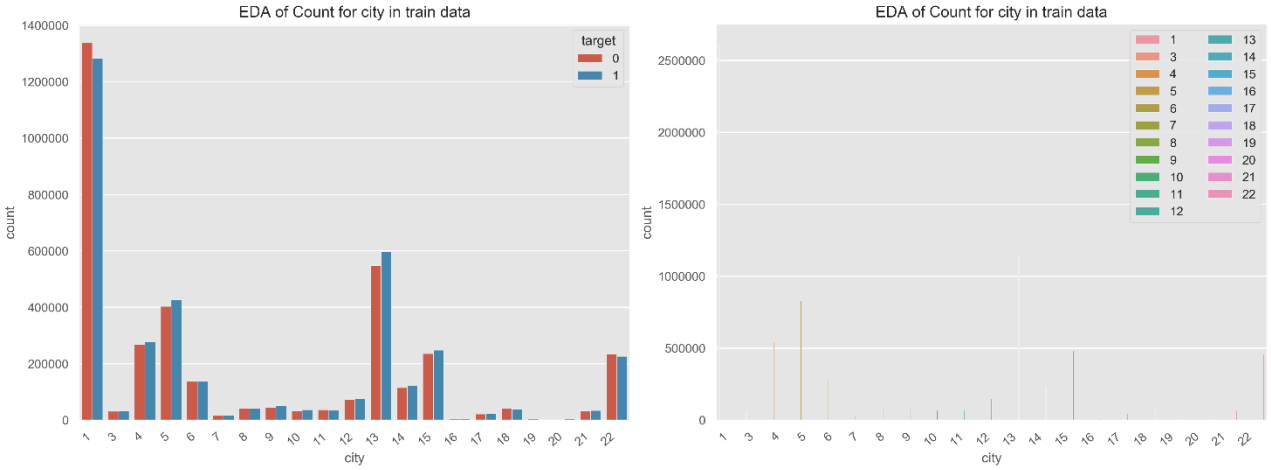


2. gender



從上圖的圓餅圖可看出遺漏值（unknown）所佔的比例高居第一，也視覺化一開始判斷遺漏值時其數量高居第二，推測應該是難以拿來成為重要變數。

3. city



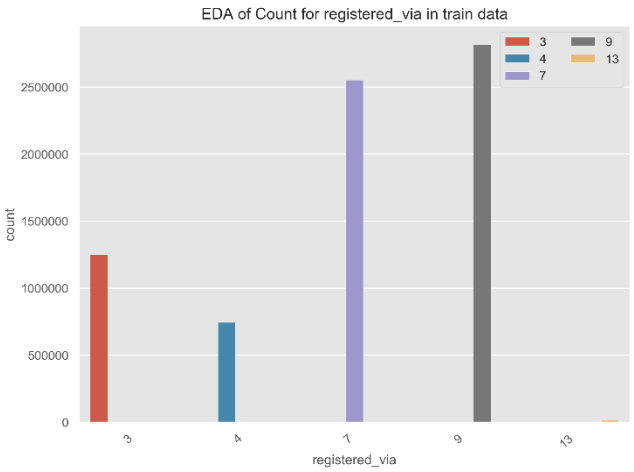
從上方右圖中可發現 city 數量最多城市代號是 1，次之為 13，第三則為 5。

再把 city 與 target 交叉比對並視覺化後發現，城市 1 的 target＝1 高居第一，將其他屬性遠遠拋在後面，且其 target＝0 的比例亦是較高的，兩者較平均高位的尚且還有城市 13 及城市 5，而其他屬性的資料筆數都不多。

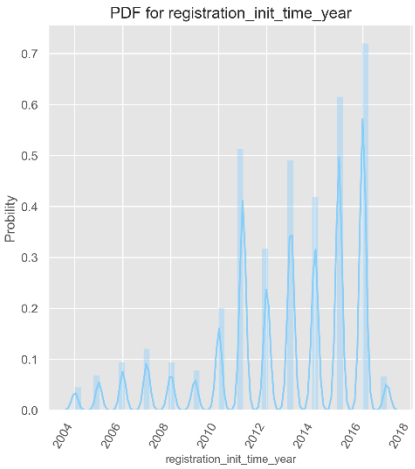
4. registered_via

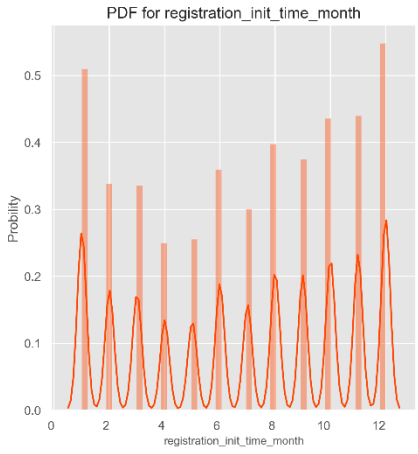
從下方右圖中可發現 registered_via 數量最多是 9，次之為 7，第三則為 3。

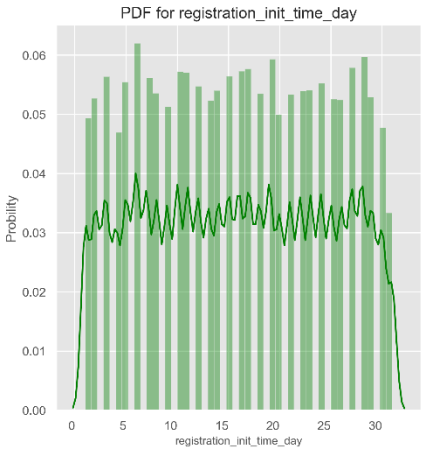
再把 registered_via 與 target 交叉比對並視覺化後發現， 9 及 7 的 target＝1 高居第一，將其他屬性遠遠拋在後面，且其 target＝0 的比例亦是較高的，兩者較平均的尚且還有 3、4，而剩餘的 13 其資料筆數不多。



5. registration_init_time







從上方左圖中可發現註冊年分較高的皆為 2011 以後，推測可能是一開始 2004 年 KKBOX 剛進市場用戶群不多，營運幾年後大力推廣註冊人數才開始攀升。且其於 2011 年推出了《Let’ s music! KKBOX 音樂誌》月刊，期望透過不同的媒體將音樂推向更廣大的市場。而每年年初所舉辦的「KKBOX 風雲榜」頒獎典禮，因擁有每年超過 50 億次點播數的公信力，更樹立了 KKBOX 在亞洲音樂業界的領導地位。

而上方中圖中可發現年初和年末的註冊月份比例相對較高，除了 4、5 月較低以外，其餘都相對平均。

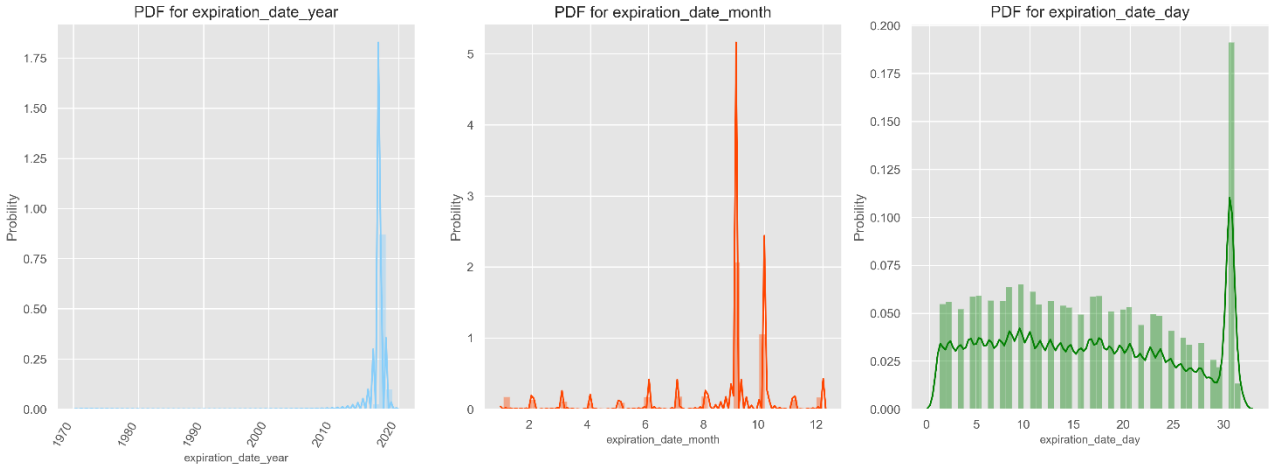
且再觀察上方左圖中可發現每日的註冊數都相對較平均，推測可能屬於較均勻的屬性。

資料來源：

KKBOX 公司介紹

https://www.kkbox.com/about/zh-tw/about

6. expiration_date

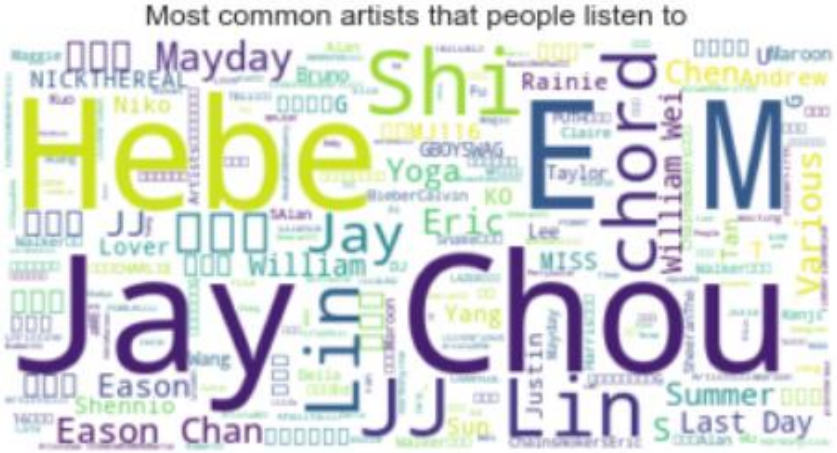


從上方左圖中可發現用戶的到期年分高峰為 2016～2017，推測可能是許多音樂串流平臺陸續升起，用戶沒有選擇續約，而是轉戰其他平臺。

而上方中圖中可發現 9 月份的到期比例最高，10 月份次之，其於皆較低。且再觀察上方左圖中可發現除了月底的到期比例最高外，推測可能是因為若是綁定電信支付的 KKBOX 會員，大多都在月底前繳費，而許多人繳完費後就不續約了。其餘日數的到期比例皆相對平均，然越靠近月底前到期比例有逐漸下降的趨勢。

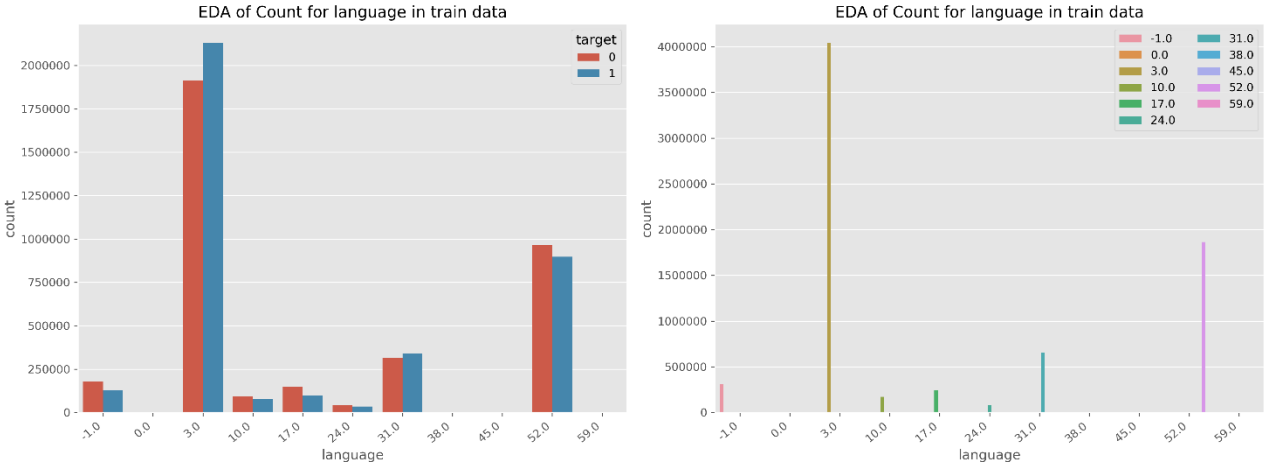
（三） Songs.csv 中的 feature

1. 熱門歌曲中的熱門歌手



上圖中的文字雲有點小瑕疵，對於中文的處理尚且沒有完善，以後有機會會更加完善且製作更多文字雲的視覺化圖形以表示熱門歌曲中的作詞作曲家、抑或是歌曲長度等等，以更精確展現資料概況。但大致上可以看出，Hebe、Jay Chou、JJ Lin、Mayday 等皆為熱門歌手，而這些歌手亦為華語歌曲中著名的代表歌手，由於前述有曾表述過 KKBOX 大多為年輕的用戶族群，且這些歌手大多為 30、40 歲年齡層的著名的熱門歌手。

2. language



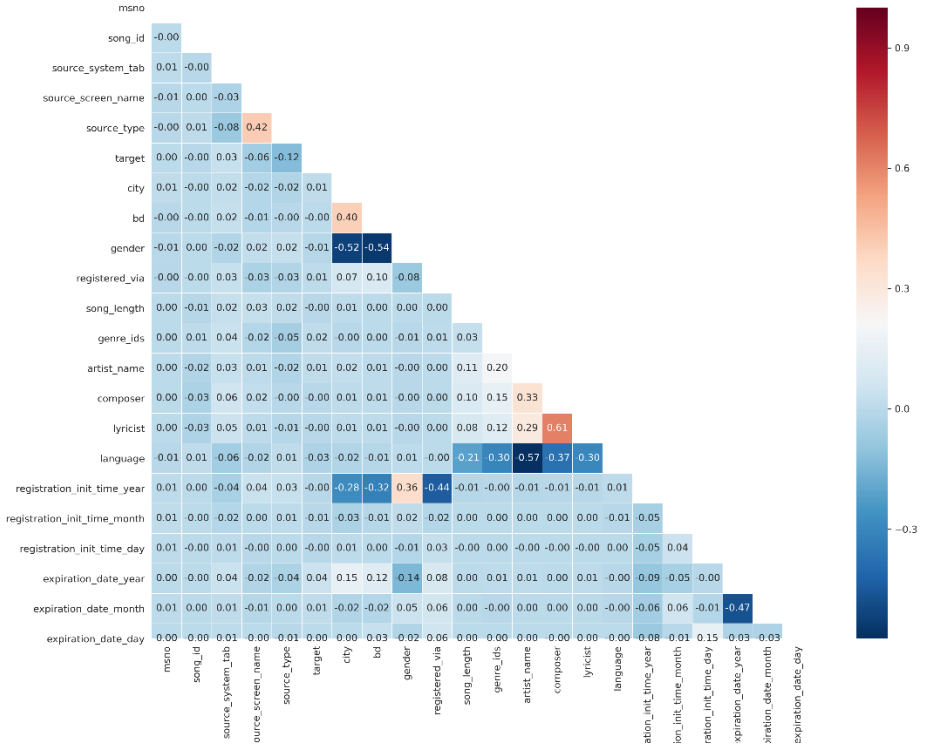
從上方右圖中可發現 language 數量最多的代號是 3.0，次之為 52.0，第三則為 31.0。

再把 language 與 target 交叉比對並視覺化後發現，3.0 的 target = 1 高居第一，將其他屬性遠遠拋在後面，且其 target＝0 的比例亦是較高的，兩者較平均的尚且還有第二位的 52.0 及第三位的 31.0，其他屬性的資料筆數都較少。

（四） 透過 EDA 進行變數初步篩選

透過前述種種的 EDA 探勘，可以發現有幾個變數也許是能用來判斷及預測的重要變數，可能對之後的模型預測有關。

在建模前先將 object 的資料格式轉成 category，以便能使用 cat.codes 的方式，將原先是類別的資料，轉換成編碼的數值型態。再將所有的屬性依照相關係數畫出熱度圖，以觀察是否有高度相關性的變數，要將其排除。



觀察上圖可發現需要排除的參數有以下兩點：

1. lyricist 和 Composer 相關係數有 0.61，故排除 lyricist，因為前述在資料勘查時，發現其 missing value 是最多的。
2. gender 和 bd 相關係數有-0.54，故排除 gender，因為前述資料勘查時，發現其 missing value 是次多的，加上 EDA 時 Male 和 Female 的比例相差不遠，且 bd 雖有 outlier 但為數不多。

```
# Drop columns
df_train_dropnew = df_train.drop(['lyricist', 'gender'], 1)
df_train_dropnew
```

executed in 2m 26s, finished 21:57:31 2021-01-19

	msno	song_id	source_system_tab	source_screen_name	source_type	target	city	bd	registered_via	song_length	genre_ids	artist_name	composer	lar
0	8158	74679	1	7	6	1	1	0	7	206471.0	285	3277	14581	52.
1	17259	223479	3	8	4	1	13	24	9	284584.0	90	31960	64996	52.
2	17259	120758	3	8	4	1	13	24	9	225396.0	90	21372	45057	52.
3	17259	23707	3	8	4	1	13	24	9	255512.0	6	27439	36700	-1.
4	8158	33308	1	7	6	1	1	0	7	187802.0	2	4472	8485	52.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7377413	4211	187659	3	8	4	1	4	26	9	267958.0	90	8749	3515	52.
7377414	18266	354712	6	15	8	0	1	0	7	227404.0	344	24013	58948	52.
7377415	18266	51546	6	15	8	1	1	0	7	258298.0	344	34937	67637	3.0
7377416	1216	102430	0	5	6	1	5	0	9	524146.0	219	4380	64996	52.
7377417	1216	123573	0	5	6	1	5	0	9	254026.0	218	3867	51420	52.

7377418 rows × 20 columns

排除前述的 lyricist 和 gender 後，df_train_dropnew 資料集有 20 個欄位（df_train 的 22 個欄位 - 2 個欄位），並將其輸出成 csv 保存。

且再將將 test 資料集準備至訓練模型前，df_test_dropnew 資料集有 20 個欄位（df_test 的 22 個欄位 - 2 個欄位），並將其輸出成 csv 保存。

```
# Drop columns
df_test_dropnew = df_test.drop(['lyricist', 'gender'], 1)
df_test_dropnew
```

executed in 17.9s, finished 19:51:09 2021-01-19

	id	msno	song_id	source_system_tab	source_screen_name	source_type	city	bd	registered_via	song_length	genre_ids	artist_name	composer	lang
0	0	12934	122191	3	8	3	1	0	7	224130.0	300	24890	33218	3.0
1	1	12934	217907	3	8	3	1	0	7	320470.0	306	24743	47677	3.0
2	2	712	37385	0	22	9	1	0	4	315899.0	158	21878	43194	17.0
3	3	1383	224360	5	16	7	3	30	9	285210.0	306	20718	39607	52.0
4	4	1383	85597	5	16	7	3	30	9	197590.0	429	21761	30731	-1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2556785	2556785	14024	212075	0	11	6	13	41	9	247640.0	100	19201	2559	52.0
2556786	2556786	18800	78127	0	22	6	13	24	9	197067.0	300	26483	46255	3.0
2556787	2556787	18800	219419	0	22	6	13	24	9	212950.0	300	23458	37581	3.0
2556788	2556788	18800	121326	0	22	6	13	24	9	164414.0	306	26764	44009	3.0
2556789	2556789	18800	154308	0	22	6	13	24	9	231552.0	300	23475	45658	3.0

2556790 rows × 20 columns

比較上述兩資料集後發現 df_test_dropnew 比 df_train_dropnew 多一個 id 欄位，而 df_train_dropnew 比 df_test_dropnew 多一個 target 欄位。故建模前要先把 df_test_dropnew 中的 id 欄位刪除，把 df_train_dropnew 中的 target 欄位分離出來變成單純的 target 欄位以及 df_train_droptarget 資料集，以利後續動作進行。

四、 建模前預先準備工作（資料切分與標準化）

（一） 分割數據集（區分訓練集與測試集）

```
# Train & Test split
target = df_train_dropnew['target']
df_train_droptarget = df_train_dropnew.drop(['target'], 1)
x_train, x_test, y_train, y_test = train_test_split(df_train_droptarget, target,
                                                    test_size = 0.2, #train:test = 8:2
                                                    random_state = 1,
                                                    stratify = target)  #stratify：依原數據y各類的比例，分配給 train 和 test
```

將訓練資料集與測試資料集以 4 比 1 的大小切割 df_train_droptarget 資料集與 target 矩陣，隨機種子採用 1，且隨機種子改變，結果多少會產生變動。

（二） 標準化

```
sc = StandardScaler()
sc.fit(x_train) #只利用training set訓練模型，故只擬合training set
x_train_std = sc.transform(x_train)
x_test_std = sc.transform(x_test)
```

由於為了消除特徵間單位和尺度差異的影響，以對每維度的特徵同等看待，需要對特徵進行標準化，而標準化後的資料可以提升模型的收斂速度及精準度。

（資料來源：為什麼要做特徵歸一化/標準化？https://codingnote.cc/zh-tw/p/20480/）

五、 建模、預測及準確度分析

我選擇使用四種模型大類，六個預測模型來進行資料建模。分別為以下六個模型，且再針對模型結果進行簡單地分析。

1. Logistic Regression
2. Decision Tree（Gini）
3. Decision Tree（Entropy）
4. Random Forest（Gini）
5. Random Forest（Entropy）
6. XGBoost

由於我一開始模型的結果並未先測試最優參數，直接使用最簡單的方式測試模型，參數頂多只有課堂上最基本的參數項。

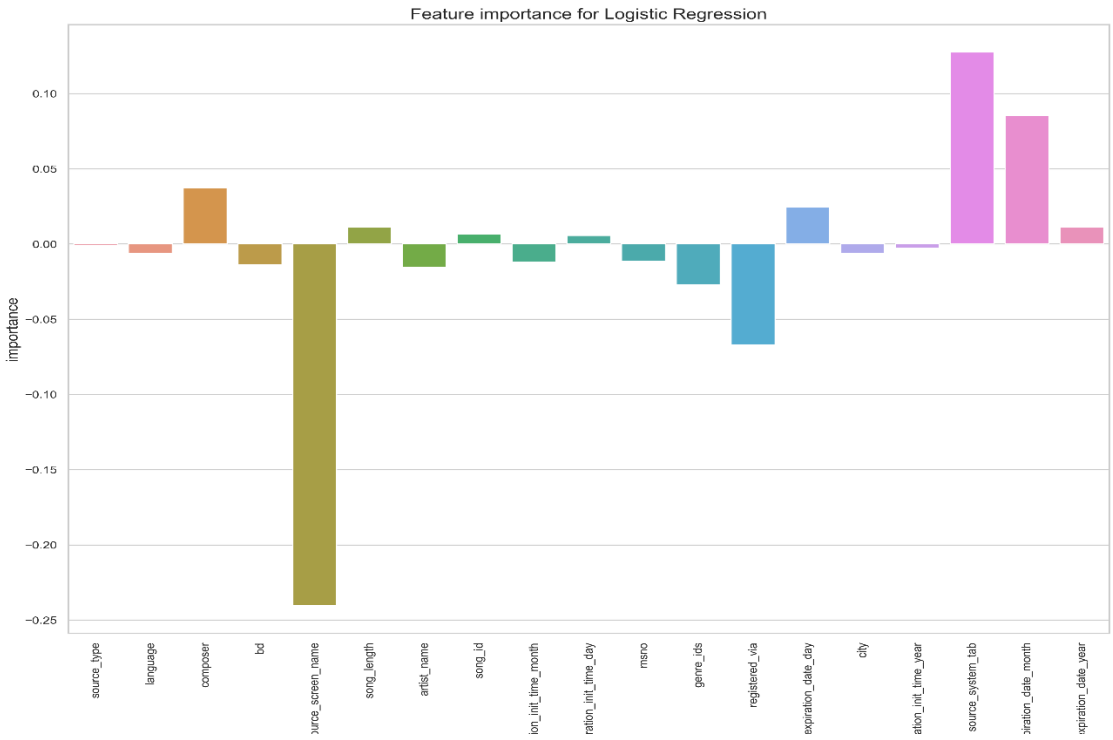
（一） Logistic Regression

1. Model 模型

```
lr = LogisticRegression(C = 100.0, random_state = 1)   #C = 1/λ
lr.fit(x_train_std, y_train)
```

其中的 C 為正則化系數 λ 的倒數，屬於 float 類型，必須是正浮點型數，默認值為 1.0，在此填 100.0 即之前課堂上初始填參數的值，沒有特地篩選過，此值若越小，其表示正則化越強。

2. Feature Importance 特徵重要性



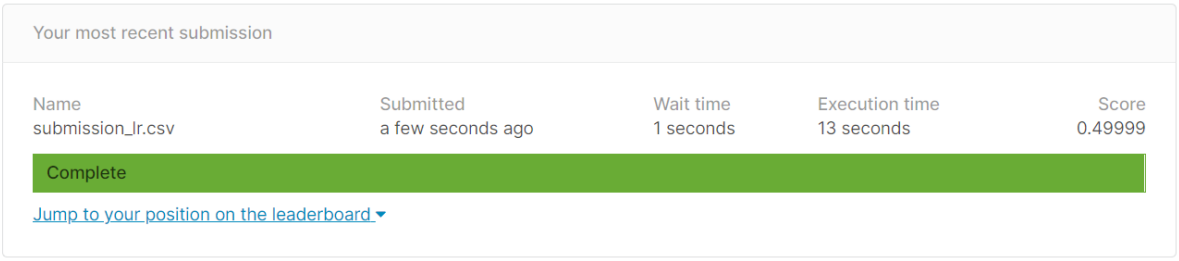
透過其 lr.coef_的方式計算出特徵的重要性，由上圖可知 Logistic Regression 中的變數重要性 source_screen_name(負)最高，次之為 source_system_tab(正)，再者為 expiration_date_month（正）。

3. 預測 Prediction 及 計算計分指標

```
**************Logistic Regression模型的準確率**************
Logistic Regression的訓練集預測正確率：59.184%
Logistic Regression的測試集預測正確率：59.133%
--->「訓練集」與「測試集」之 預測正確率--->相差0.050%
************************************************************

Logistic Regression Classification Report:
************************************************************
              precision    recall  f1-score   support

           0       0.59      0.56      0.58    732553
           1       0.59      0.62      0.61    742931

    accuracy                           0.59   1475484
   macro avg       0.59      0.59      0.59   1475484
weighted avg       0.59      0.59      0.59   1475484

************************************************************
```

由上圖可看出測試集預測準確率僅有 60%，雖與訓練集的正確率相差僅約 0.05%，再加上其分類報告中 precision 分數為 0.59、recall 為 0.62、f1-score 為 0.61，整體看起來模型測試可能尚須要加強抑或是調參數以優化模型。

4. 匯出 csv & Submission 提交為 Kaggle 的成績



分數僅有 0.49999，大約 0.5，意即猜對的比率僅有一半而已，尚需加強。

（二） Decision Tree（Gini）

1. Model 模型

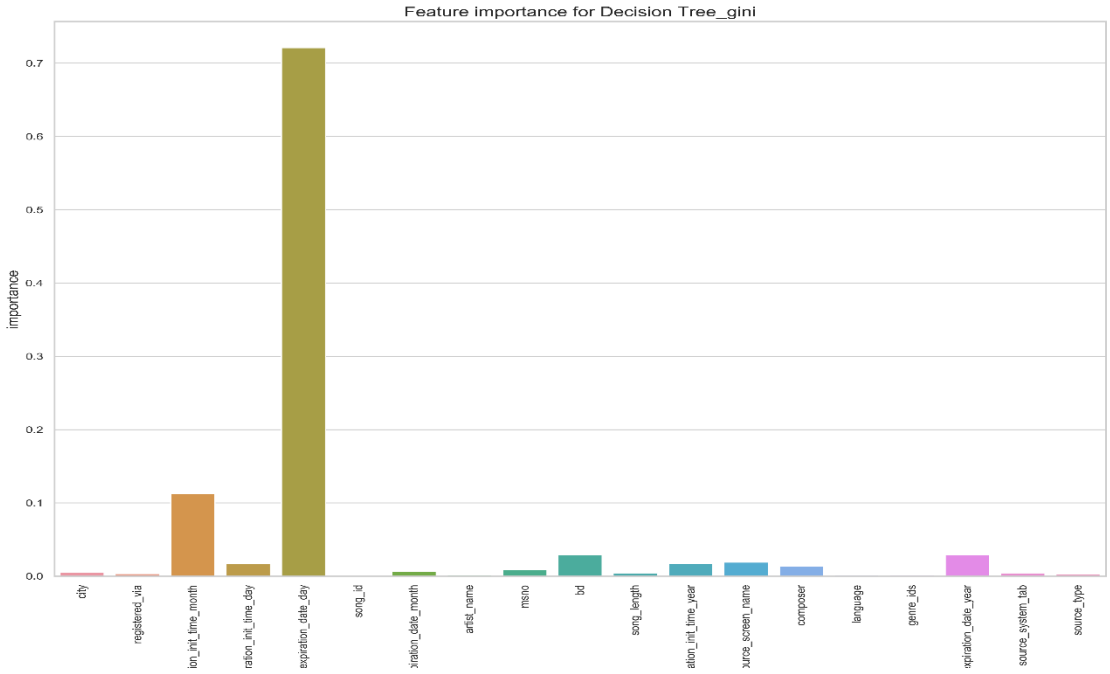
```
#Decision Tree_gini
tree_gini = DecisionTreeClassifier(criterion = 'gini', random_state = 1, max_depth = 8)
tree_gini.fit(x_train_std, y_train)
```

這裡僅初步設定最大深度（樹高）為 8，方法為 Gini，隨機種子採 1。

採用 Gini 的代表是 CART tree，其為 Classification And Regression Tree 的縮寫，從字面上可看出它兼具分類與迴歸兩種功能，同時支援分類(Classification)與數字預測（Regression），由於不限制應變數與自變數的類型，因此在使用上較具彈性，是目前最為常用的決策樹方法，故在此選擇此模型作為試驗對象。

資料來源：CH.Tseng 決策樹 Decision trees

2. Feature Importance 特徵重要性



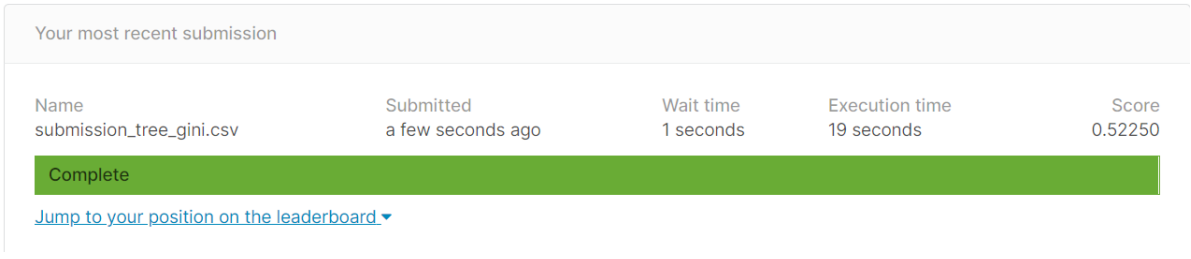
透過模型本身內含的特徵重要性函數計算出特徵的重要性，由上圖可知 Decision Tree（Gini）中的變數重要性 expiration_date_day（正）最高，次之為 registration_init_time_month（正），再者為 bd 與 expiration_date_year（正）。

3. 預測 Prediction 及 計算計分指標

```
**************Decision Tree Gini模型的準確率**************
Decision Tree Gini的訓練集預測正確率：63.219%
Decision Tree Gini的測試集預測正確率：63.133%
--->「訓練集」與「測試集」之 預測正確率--->相差0.086%
************************************************************

Decision Tree Gini Classification Report:
************************************************************
              precision    recall  f1-score   support

           0       0.63      0.63      0.63    732553
           1       0.63      0.63      0.63    742931

    accuracy                           0.63   1475484
   macro avg       0.63      0.63      0.63   1475484
weighted avg       0.63      0.63      0.63   1475484

************************************************************
```

由上圖可看出測試集預測準確率僅有 63%，雖與訓練集的正確率相差僅約 0.086%，再加上其分類報告中 precision 分數為 0.63、recall 為 0.63、f1-score 為 0.63，整體看起來模型測試可能尚須要加強抑或是調參數以優化模型。

4. 匯出 csv & Submission 提交為 Kaggle 的成績



分數僅有 0.52250，大約 0.52，意即猜對的比率僅趨近一半，尚需加強。

（三） Decision Tree（Entropy）

1. Model 模型

```
##Decision Tree_entropy
tree_entropy = DecisionTreeClassifier(criterion = 'entropy', random_state = 1, max_depth = 8)
tree_entropy.fit(x_train_std, y_train)
```

這裡僅初步設定最大深度（樹高）為 8，方法為 Entropy，隨機種子採 1。

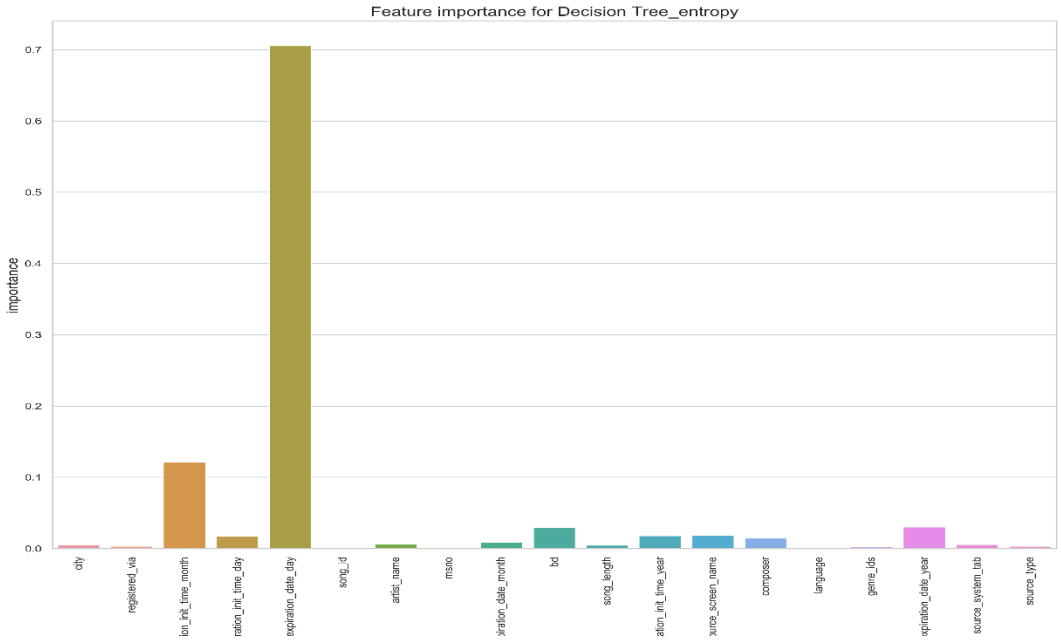
Entropy（熵）就是用來度量資訊量的方法，也就是假設有一個資料集很純淨，無法用其他屬性進行分類，以及有一個資料集較雜亂，可以用很多屬性進行分類，那前者純淨的資料集，假設資料都一致，其 entropy = 0，反之雜亂的資料集如果各有一半不同，那 entropy = 1。

對於資訊量大小和其不確定性有直接的關係，要搞清楚一件非常非常不確定的事情，或者是一無所知的事情，需要了解大量資訊（資訊量的度量）就是等於不確定性的多少。故在此選擇此模型作為試驗對象。

資料來源：

CH.Tseng 決策樹 Decision trees、【深度學習基礎-03】決策樹演算法-熵如何計算舉例

2. Feature Importance 特徵重要性



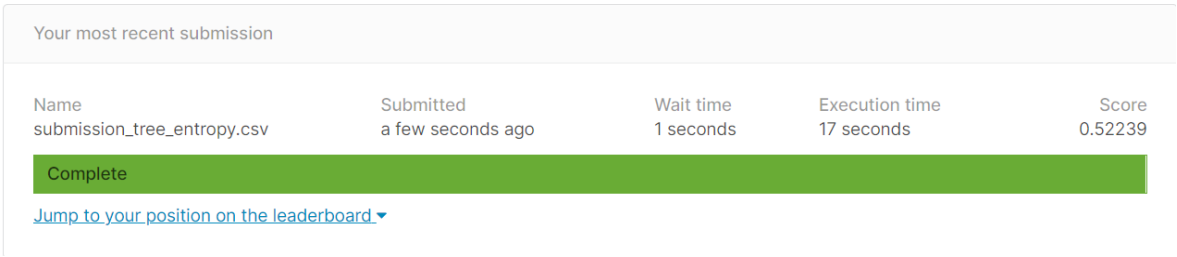
透過模型本身內含的特徵重要性函數計算出特徵的重要性，由上圖可知 Decision Tree（Entropy）中的變數重要性 expiration_date_day（正）最高，次之為 registration_init_time_month（正），再者為 bd 與 expiration_date_year（正）。結果與 Decision Tree（Gini）相同

3. 預測 Prediction 及 計算計分指標

```
**************Decision Tree Entropy模型的準確率**************
Decision Tree Gini的訓練集預測正確率：63.205%
Decision Tree Gini的測試集預測正確率：63.119%
--->「訓練集」與「測試集」之 預測正確率--->相差0.086%
************************************************************

Decision Tree Entropy Classification Report:
*********************************************************
              precision    recall  f1-score   support

           0       0.63      0.63      0.63    732553
           1       0.63      0.63      0.63    742931

    accuracy                           0.63   1475484
   macro avg       0.63      0.63      0.63   1475484
weighted avg       0.63      0.63      0.63   1475484

*********************************************************
```

由上圖可看出測試集預測準確率僅有 63%，雖與訓練集的正確率相差僅約 0.086%，再加上其分類報告中 precision 分數為 0.63、recall 為 0.63、f1-score 為 0.63，整體看起來模型測試可能尚須要加強抑或是調參數以優化模型，結果與 Decision Tree（Gini）相近。

4. 匯出 csv & Submission 提交為 Kaggle 的成績



分數僅有 0.52239，大約 0.52，意即猜對的比率僅趨近一半，尚需加強，結果比 Decision Tree（Gini）稍低一點。

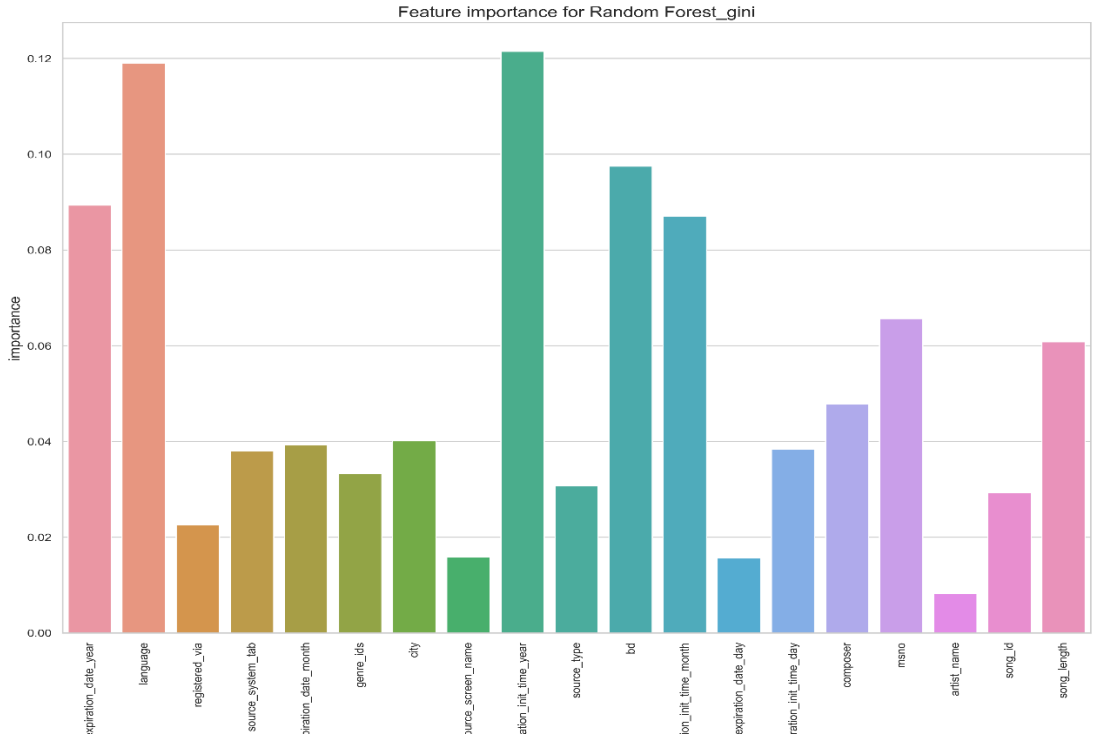
（四） Random Forest（Gini）

1. Model 模型

```
#Random Forest_gini
forest_gini = RandomForestClassifier(criterion = 'gini', n_estimators = 50, random_state = 1)
forest_gini.fit(x_train_std, y_train)
```

這裡僅初步設定有 50 棵決策樹，方法為 Gini，隨機種子採 1。

2. Feature Importance 特徵重要性



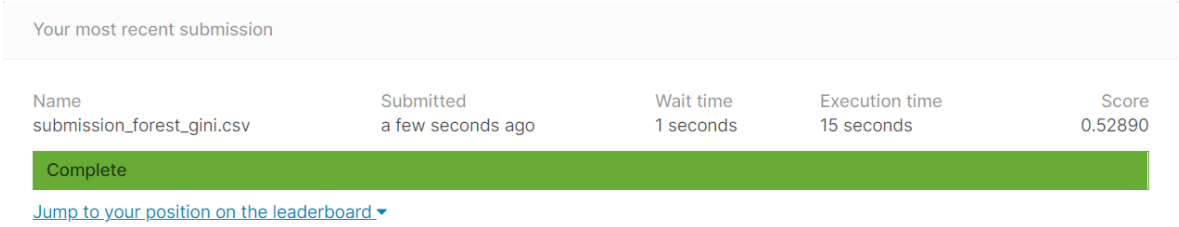
透過模型本身內含的特徵重要性函數計算出特徵的重要性，由上圖可知 Random Forest（Gini）中的變數重要性 registration_init_time_year（正）最高，次之為 language（正），再者為 bd（正）。

3. 預測 Prediction 及 計算計分指標

```
**************Random Forest Gini模型的準確率**************
Random Forest Gini的訓練集預測正確率：99.983%
Random Forest Gini的測試集預測正確率：72.595%
--->「訓練集」與「測試集」之 預測正確率--->相差27.387%
************************************************************

Random Forest Gini Classification Report:
*********************************************************
              precision    recall  f1-score   support

           0       0.72      0.74      0.73    732553
           1       0.73      0.71      0.72    742931

    accuracy                           0.73   1475484
   macro avg       0.73      0.73      0.73   1475484
weighted avg       0.73      0.73      0.73   1475484

*********************************************************
```

由上圖可看出測試集預測準確率有 73%，與訓練集的正確率相差約 27.387%，再加上其分類報告中 precision 分數為 0.73、recall 為 0.71、f1-score 為 0.72，整體看起來模型測試可能尚須要加強抑或是調參數以優化模型，因為測試集的準確率有達到近 100%。

4. 匯出 csv & Submission 提交為 Kaggle 的成績



分數僅有 0.52890，大約 0.53，意即猜對的比率僅趨近一半，尚需加強，結果比 Decision Tree 稍高一點。

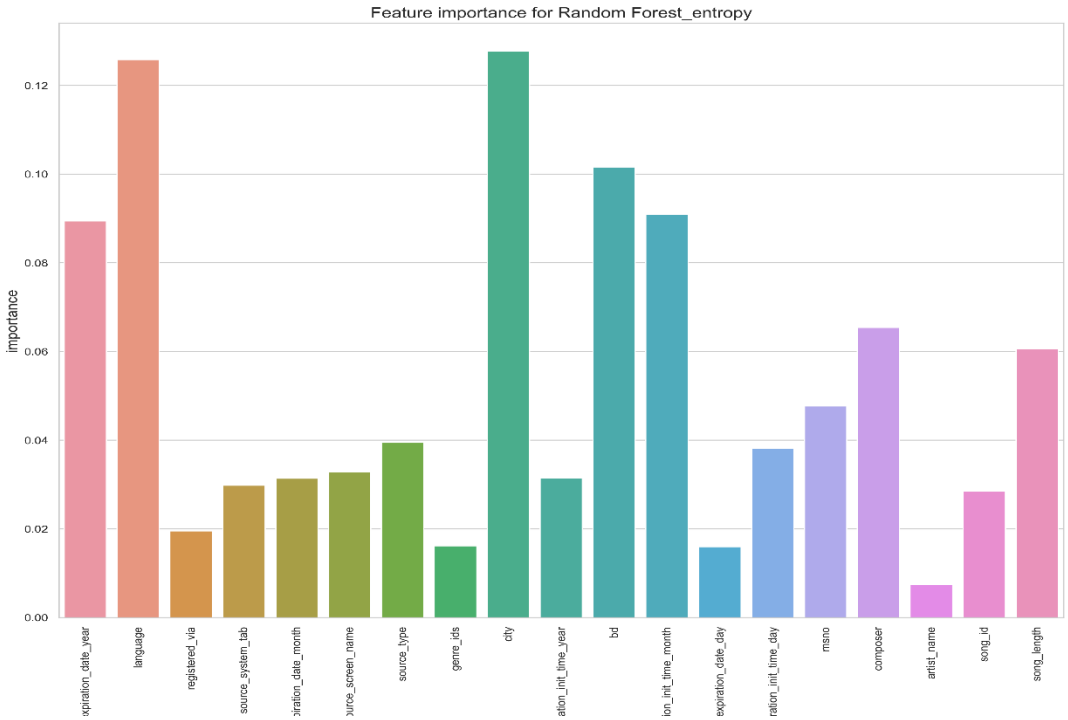
（五） Random Forest（Entropy）

1. Model 模型

```
#Random Forest_entropy
forest_entropy = RandomForestClassifier(criterion = 'entropy', n_estimators = 50, random_state = 1)
forest_entropy.fit(x_train_std, y_train)
```

這裡僅初步設定有 50 棵決策樹，方法為 Entropy，隨機種子採 1。

2. Feature Importance 特徵重要性



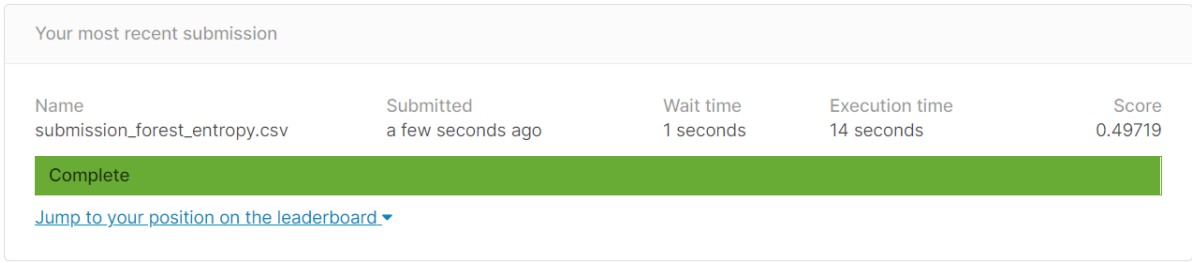
透過模型本身內含的特徵重要性函數計算出特徵的重要性，由上圖可知 Random Forest（Entropy）中的變數重要性 city（正）最高，次之為 language（正），再者為 bd（正）。與 Random Forest（Gini）中最重要的變數產生相異性。

3. 預測 Prediction 及 計算計分指標

```
***************Random Forest Entropy模型的準確率***************
Random Forest Entropy的訓練集預測正確率：99.982%
Random Forest Entropy的測試集預測正確率：72.688%
--->「訓練集」與「測試集」之 預測正確率--->相差27.294%
************************************************************

Random Forest Entropy Classification Report:
*******************************************************
              precision    recall  f1-score   support

           0       0.72      0.74      0.73    732553
           1       0.74      0.71      0.72    742931

    accuracy                           0.73   1475484
   macro avg       0.73      0.73      0.73   1475484
weighted avg       0.73      0.73      0.73   1475484

*******************************************************
```

由上圖可看出測試集預測準確率有達到 73%，與訓練集的正確率相差約 27.294%，再加上其分類報告中 precision 分數為 0.74、recall 為 0.71、f1-score 為 0.72，整體看起來模型測試可能尚須要加強抑或是調參數以優化模型，因為測試集的準確率有達到近 100%。

4. 匯出 csv & Submission 提交為 Kaggle 的成績



分數僅有 0.49719，大約 0.50，意即猜對的比率僅趨近一半，尚需加強，結果比 Random Forest（Gini）低大約 0.53，整體表現看起來目前為倒數第二。

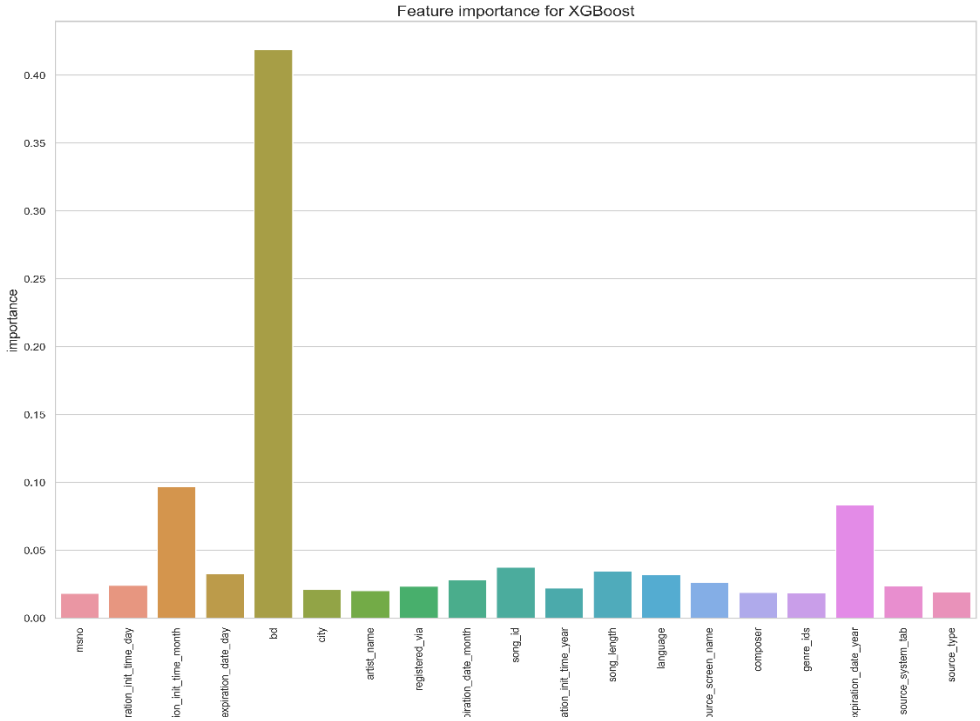
（六） XGBoost

1. Model 模型

```
xgb = XGBClassifier()
xgb.fit(x_train_std, y_train)
```

這裡沒有設定任何的參數，直接用最原始的 XGBoost 媒合測試，應該準確率也不會太高，後續可能需要對參數進行更加精確的調整。

2. Feature Importance 特徵重要性



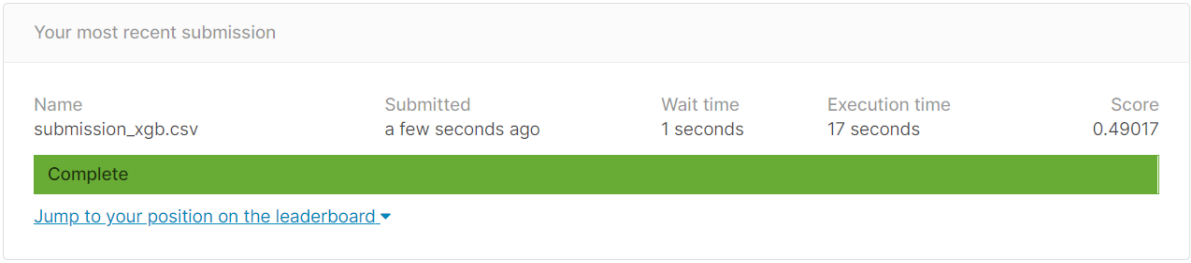
透過模型本身內含的特徵重要性函數計算出特徵的重要性，由上圖可知 XGBoost 中的變數重要性 bd(正)最高，次之為 registration_init_time_month(正)，再者為 expiration_date_year（正）。

3. 預測 Prediction 及 計算計分指標

```
**************XGBoost模型的準確率**************
XGBoost的訓練集預測正確率：65.920%
XGBoost的測試集預測正確率：65.745%
--->「訓練集」與「測試集」之 預測正確率--->相差0.176%
************************************************************

XGBoost Classification Report:
************************************************************
              precision    recall  f1-score   support

           0       0.66      0.63      0.64    732553
           1       0.65      0.69      0.67    742931

    accuracy                           0.66   1475484
   macro avg       0.66      0.66      0.66   1475484
weighted avg       0.66      0.66      0.66   1475484

************************************************************
```

由上圖可看出測試集預測準確率僅有 66%，僅與訓練集的正確率相差約 0.176%，再加上其分類報告中 precision 分數為 0.65、recall 為 0.69、f1-score 為 0.67，整體看起來模型測試可能尚須要加強抑或是調參數以優化模型。

4. 匯出 csv & Submission 提交為 Kaggle 的成績



分數僅有 0.49017，大約 0.50，意即猜對的比率僅趨近一半，尚需加強，整體表現看起來目前最差的結果，可能是因為完全都沒有設定參數，即選用最原始的方始判斷其模型原先的準確率。

（七） 統整六個模型的各項數值

1. 模型準確率

模型名稱	訓練資料集	測試資料集
Logistic Regression	59.184%	59.133%
Decision Tree（Gini）	63.219%	63.133%
Decision Tree（Entropy）	63.205%	63.119%
Random Forest（Gini）	99.983%	72.595%
Random Forest（Entropy）	99.982%	72.688%
XGBoost	65.920%	65.745%

2. 模型的評估分數

模型名稱	Precision	Recall	F1-score	F1-score 排名
Logistic Regression	0.59	0.62	0.61	4
Decision Tree（Gini）	0.63	0.63	0.63	3
Decision Tree（Entropy）	0.63	0.63	0.63	3
Random Forest（Gini）	0.73	0.71	0.72	1
Random Forest（Entropy）	0.74	0.71	0.72	1
XGBoost	0.65	0.69	0.67	2

六、 未來改善及心得

由於時間緊迫，加上第一次做 Kaggle 競賽，再加上由於是自己一個人做，所以基本上真的是從頭開始研究，觀察，詢問很多人，才慢慢建立自己的架構和想法，雖然很多內容都是仿照許多網路上的資料，但自己獨立寫完一整個程式碼，真的很感動，也很辛苦，畢竟我開始寫程式不到一年而已，雖然可能結果並不好，但我已經很滿足了。

以下為未來要改善的部分，希望可以再將內容表現得更加完善。

1. 針對模型進行選擇並調整參數。

   因為每個模型都要執行好久，所以沒有機會可以選擇參數，加上一直想用餐數選擇來最佳化模型，希望有機會可以做到。

2. 嘗試使用不同的模型，例如：SVM

   嘗試不同的情況下使用不同的建模法，應該將資料的情境和特性再次的劃分，並以情境去適用不同的演算法，而非將所有資料套到同一個演算法，也許會有不錯的效果。

七、 參考資料

1. 淺談何謂探索式資料分析 https://ithelp.ithome.com.tw/articles/10213384
2. KKBOX 歌曲推薦系統 – 一定要配溫開水 https://wenwender.wordpress.com/2019/04/03/kkbox-%E6%AD%8C%E6%9B%B2%E6%8E%A8%E8%96%A6%E7%B3%BB%E7%B5%B1/
3. KKBOX 公司介紹 https://www.kkbox.com/about/zh-tw/about
4. 為什麼要做特徵歸一化/標準化？https://codingnote.cc/zh-tw/p/20480/