109智慧化企業整合

預測KKBOX客戶流失率

第2組
黃怡菁 李旖庭 黃荻雅 黃浩銓

OUTLINE

01. SCENARIO

02. DATA PREPROCESSING

03. MODEL

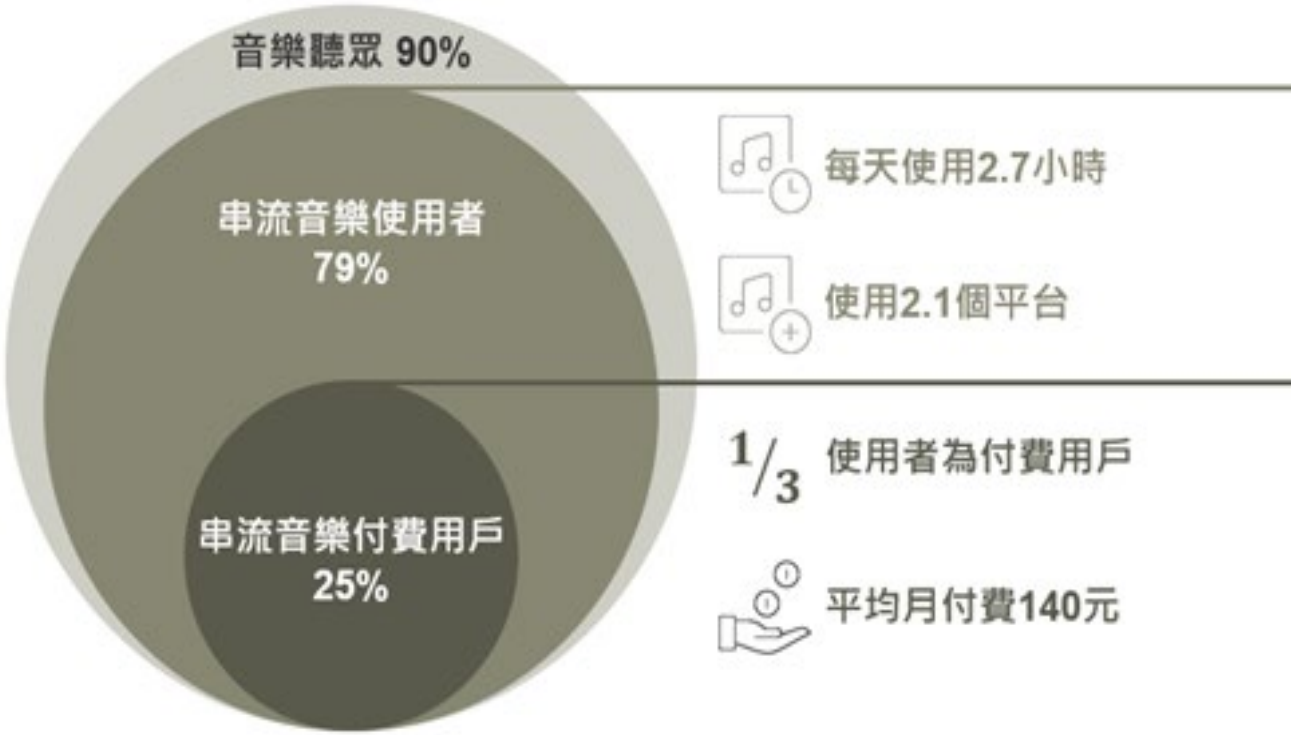
04. ANALYSIS

05. CONCLUSION

01 SCENARIO

PROBLEM DEFINITION

Scenario
Background



“線上串流音樂平台的使用率有79%，觀察不同年齡的使用率後，發現20世代超過八成有使用線上串流音樂平台，比率明顯較其他年齡層更高，使用率高於整體10%。”

“KKBOX作為台灣串流音樂的領導品牌，無論在知名度（66%）、目前使用普及（25%）或付費普及率（16%）都最高，換算起來64%的用戶皆為付費用戶，居各音樂串流平台之冠。”

Scenario

Problem Definition — 5W1H

WHAT

KKBOX訂閱用戶預測

WHERE

KKBOX行銷部門

WHEN

KKBOX欲評估顧客訂閱率時

WHY

訂閱用戶流失率難以預測，卻至關重要

WHO

各大影音串流平台 ex:KKBOX

HOW

資料分析、深度學習、機器學習

02

DATA PREPROCESSING

Data-Preprocessing
Data Resource

- 來自於Kaggle，數據建模和數據分析競賽平台
- 企業和研究者可發布數據
- 統計學家和數據挖掘專家已次資料為基礎，進行競賽以產生最好的模型
- 使用Kaggle上的「KKBox’s Churn Prediction Challenge」為數據庫
- 總共有886500筆資料(用戶)。由KKBOX提供用戶使用情況以此資料來預測此用戶在到期日後30天后是否會繼續訂閱，如果沒有，則判定為流失客戶

Data-Preprocessing

Data Meaning

特徵	說明
msno	使用者id
is churn	用戶是否會在會員到期日後30天內繼續訂閱，以此定義客戶是否流失
data	用戶使用KKBOX聽音樂的日期
num_25	每首歌只播放完整時間的前25%的數量
um_50	每首歌只播放完整時間的25%~50%的數量
um_75	每首歌只播放完整時間的50%~75%的數量
um_985	每首歌只播放完整時間的70%~98.5%的數量
um_100	每首歌完整播放的數量
um_unq	總共有多少不同的歌曲是被用戶收聽的
Total secs	總播放時間
city	城市
bd	年紀
gender	性別
registered_via	註冊方式
registration_init_time	註冊時間
expiration_date	到期時間

Data-Preprocessing

Data Content

```
%bq tables list
```

- iie-project2.kk_data.members_v3
- iie-project2.kk_data.sample_submission_v2
- iie-project2.kk_data.sample_submission_zero
- iie-project2.kk_data.train
- iie-project2.kk_data.train_v2
- iie-project2.kk_data.transactions
- iie-project2.kk_data.transactions_v2
- iie-project2.kk_data.user_logs
- iie-project2.kk_data.user_logs_v2

原始資料分別存在不同資料表裡，首先我們先將擁有相同資料欄位的資料表合併在一起

```
%bq tables list
```

- iie-project2.kk_data.members_v3
- iie-project2.kk_data.sample_submission_zero
- iie-project2.kk_data.train
- iie-project2.kk_data.transactions
- iie-project2.kk_data.user_label_201703
- iie-project2.kk_data.user_logs

因為_V2是後續更新的資料，所以我們將其合併成一個以利後續我們分析。

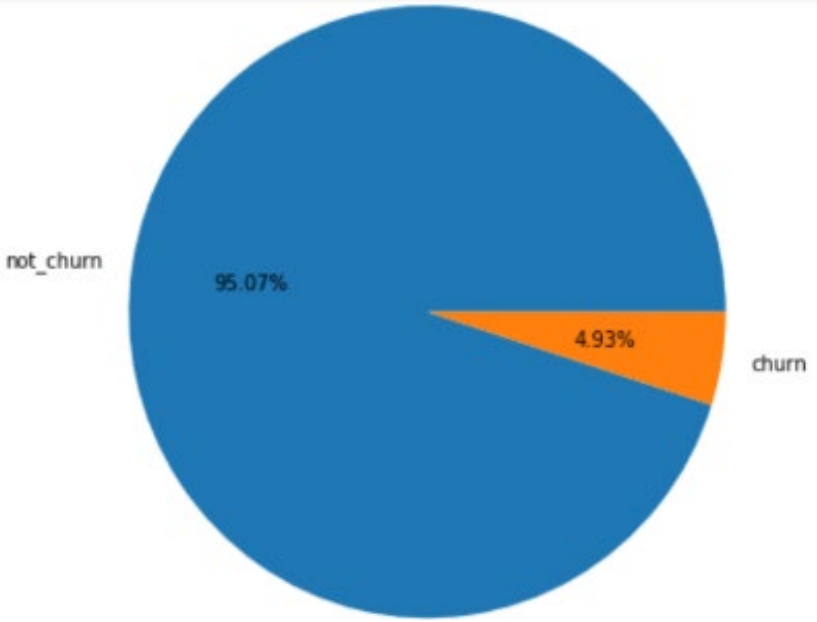
Data-Preprocessing
Data Analysis

```
[ ] churn_df.shape

(886500, 2)
```

```
churn_df.head()
```

	msno	is_churn
0	++4RuqBw0Ss6bQU4oMxaRlbBPoWzoEilZaxPM04Y4+U=	False
1	+/HS8LzrRGXolKbxRzDLqrmwuXqPOYixBlPXkyNcKNI=	False
2	+/g9O3USecrC8npzaFHXW/2XJ7fB80SineiUoCg7M6o=	False
3	+/namlXq+u3izRjHCFJV4MgqcXcLidZYszVsROOq/y4=	False
4	+0/X9tkmyHyet9X80G6GTrDFHnJqvai8d1ZPhayT0os=	False



先從User_Label看總共有886500筆資料，並且進一步查看其用戶流失分布

可以發現流失的用戶很少，這樣使得我們的資料集不平衡(沒有流失的用戶很明顯大於流失的用戶) 我們可能需要對資料集調整訓練時的類別權重，或是過濾掉一些資料

Data-Preprocessing
Data Analysis

(rows: 410502905, iie-project2.kk_data.user_logs)

接著再看User_Logs資料表，總共有410502905筆資料

	not_null_date	not_null_25	not_null_50	not_null_75	not_null_985	not_null_100	not_null_nuq	not_null_total_secs
0	410502905	410502905	410502905	410502905	410502905	410502905	410502905	410502905

進一步檢查是否存在空值，發現沒有空值，代表所有列都是有數值不為0

	msno	date	num_25	num_50	num_75	num_985	num_100	num_unq	total_secs
1	8n5yuVdv8qR4aHr1hUx1FXoIuxpZluW+kR/d0ounuYA=	20,160,512	24	2	1	1	53	61	15,019.365
2	d2GtIMHD45Ri5/Ksa86X4FYWCZMM3fZ4WRDKjbhp15A=	20,160,205	81	2	4	3	34	74	9,600.676
3	d2GtIMHD45Ri5/Ksa86X4FYWCZMM3fZ4WRDKjbhp15A=	20,161,124	37	4	4	2	48	83	12,778.431
4	d2GtIMHD45Ri5/Ksa86X4FYWCZMM3fZ4WRDKjbhp15A=	20,170,114	113	22	8	3	40	163	14,347.198
5	kb3qHtlz+K4Ume8TF4FQi9xwrTZqzFFvBDZsdYvyQ0A=	20,161,217	32	1	0	1	30	58	7,391.719
6	B/eZk3P+A98+vport4EL6KBRhYiioS+F1uVJSGmAUGw=	20,150,720	22	2	0	2	38	52	10,062.741
7	ysLUp9Ebqx3RrCNmZAO5myW7kDZQafvyg7+Ge6lbG3Y=	20,150,717	38	6	5	4	66	87	20,265.215
8	ysLUp9Ebqx3RrCNmZAO5myW7kDZQafvyg7+Ge6lbG3Y=	20,160,318	36	4	2	4	97	61	24,174.545
9	65MC0qTNLb/tG6fPv0IN7AzLqma4IkDHe1SEB8TedA8=	20,150,411	42	24	13	6	85	131	26,109.762
10	aDyAkp8ZPYJUAVI5QZ9oe1/2Ub1iDbq1Z9B6lBaTGRk=	20,150,107	23	12	3	7	60	90	19,544.564

此資料表裡的有效資料數遠大於用戶數886500，又發現mson用戶名稱是有重複出現的，代表這張資料表紀錄的是當天用戶聽歌情形

Data-Preprocessing
Data Analysis

	feature	min	Q1	Q2	Q4	max
1	num_25	0.0	0.0	1.0	2.0	1710.0
0	num_50	0.0	0.0	2.0	7.0	18798.0
2	num_75	0.0	0.0	0.0	1.0	1690.0
3	num_985	0.0	0.0	0.0	1.0	2747.0
4	num_100	0.0	6.0	17.0	38.0	42004.0
5	num_unq	1.0	8.0	19.0	40.0	4925.0
6	total_sec	-9223372036854776.0	1894.0	4636.0	10228.0	9223372036854776.0

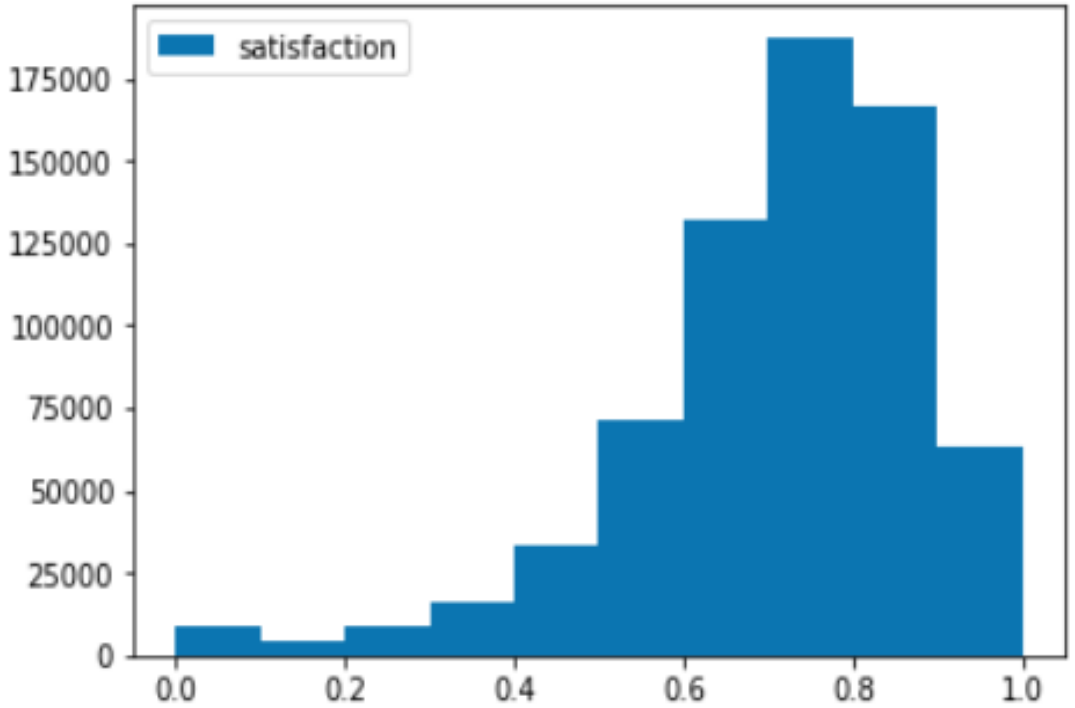
從total_secs的結果可以看到有異常值的出現，顯示我們需要進行資料清理，而清理資料需要評估我們是否會用到這些特徵

使用者聽歌25%, 50%, 75%的比例不高，平均數分別為1, 2, 0，我們可以推測其卡歌率不高，有可能是使用者們滿意自己所聽的音樂

此資料集據官方描述是以一天為一筆，我們可以粗估使用者每天大約會聽20首歌。(平均一天聽4636s，一首歌4 min(240 s)，平均一天聽4636 / 240 ≒ 20首)

這裡的最大值都是值得懷疑的，官方文件寫說user_logs為每日紀錄，我們觀察num_100這一列，一首歌4分鐘也就是240秒，平均來說一天(86400)最多只有可能聽到360首完整撥放的歌曲

- EDA(Exploratory Data Analysis)
- 以is_churn做區隔分別畫出day_listen和user_latent_satisfaction的分布



左圖為使用者的滿意度分布

在不會流失的使用者中，大多數滿意度較高

➢ 觀察訓練集和測試集的Missing value

```
train_df.isnull().sum()
```

```
msno                 0
is_churn             0
day_listen      153632
satisfaction    153632
dtype: int64
```

```
test_df.isnull().sum()
```

```
msno                 0
is_churn             0
day_listen           0
satisfaction    599274
dtype: int64
```

我們發現如果只抽取一個月，缺失值比例會比較高，依序兩個月，三個月，缺失值比例會遞減，這對於我們機器學習模型來說，抽取越多個月表示資料將會越完整

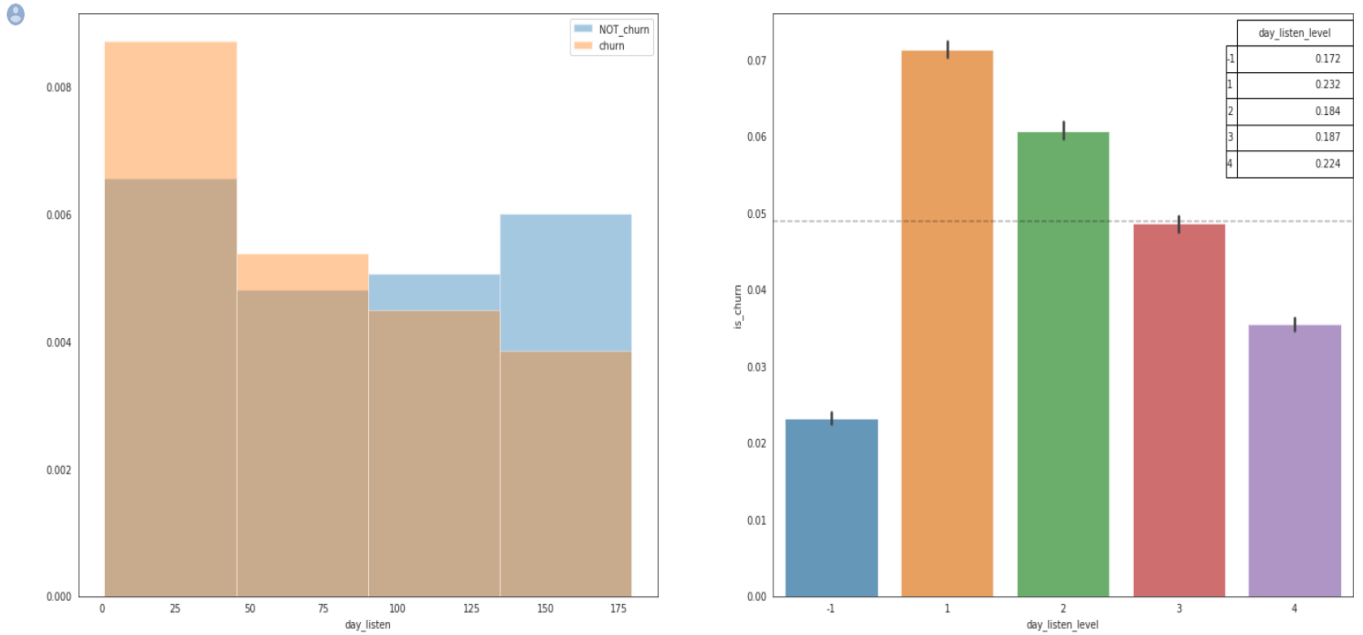
太久遠的資料對於目標函數的參考價值也會降低，因此這裡有兩個之間作取捨，使用驗證集來測試抽取時間的參數應該幾個月比較好，最終測試得到的結果是抽取6個月的數據

將缺失值以-1填入，做為一個特徵類別(將其看成一個新的特徵)

Data-Preprocessing
Data Cleaning

- **觀察訓練集和測試集的Missing value**

```
    ax1.legend()
# barplot with churn_rate plus df_percent table
    sns.barplot(x='day_listen_level', y='is_churn', data=df,ax=ax2,alpha=.75)
    ax2.axhline(0.049,linestyle='dashed',c='black',alpha=.3)
    table(ax2, df_percent,bbox=[0.80, 0.72, 0.2, 0.27])
```



左圖為聽歌天數(day_listen)的分布圖，橘色表示流失。

右圖為區塊內的流失率，其中-1為缺失值。

從右圖中看出隨著聽歌天數(day_listen)越多，則流失的機率越低

右圖將day_listen分為5個level，再分別去計算其比例，其數量佔比最少都有17%以上，這使得流失率的準確度是可以相信的

03

MODEL

Random Forest & XGBoost

Model Structure
Random Forest

➢ **參數設定**

```
print(df_train.columns)
print(df_sub.columns)
```

```
Index(['msno', 'is_churn', 'day_listen', 'user_latent_satisfactio
n',
       'day_listen_level'],
      dtype='object')
Index(['msno', 'is_churn', 'day_listen', 'user_latent_satisfactio
n',
       'day_listen_level'],
      dtype='object')
```

➢ **模型建立**

```
model = RandomForestClassifier(random_state=2,n_estimators=300,
                               min_samples_split=0.05,n_jobs=-1,class_weight={0 :0.45,1 :0.55})
```

- Random state：用來控制forest生成的模式，使它不會固定只生成一種tree
- n_estimators：最大迭帶次數
- min_samples_spl：內部節點再劃分所需最小樣本數
- n_job：應用於bagging
- class_weigh：每個label的權重
- Train(訓練集) : Test(測試集) = 8 : 2

Model Structure
Random Forest

➢ **完整模型**

```
def model_training_rf(training_data,testing_data):
    # splits train and validation set
    X = training_data.drop(labels=['msno','is_churn'],axis=1)
    Y = training_data['is_churn']
    X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size=0.2,random_state = 2)
    # Training ~ 01:45s
    model = RandomForestClassifier(random_state=2,n_estimators=300,
                                                    min_samples_split=0.05,n_jobs=-1,class_weight={0 :0.45,1 :0.55})
    model.fit(X_train,Y_train)

    # caculating E_val

    model_probs = model.predict_proba(X_val)
        # [:,1] to show the prob to is_churn = 1
    model_val_score = log_loss(Y_val,model_probs[:,1])

    # predict on testing set
    model_pred_testing_set = model.predict_proba(testing_data.drop(labels=['msno','is_churn'],axis=1))
    model_pred_testing_set = model_pred_testing_set[:,1] # take out the prob if is_churn = 1
    submission = pd.DataFrame({"msno": testing_data.msno})
    submission.insert(1,column='is_churn',value=model_pred_testing_set)

    return model, model_val_score, submission
```

Model Structure
Random Forest

- 分別對不同的資料作建模

```
[ ]  #沒切分的資料建模
     orginal_day_listen_model,  orginal_day_listen_val_score,\
     orginal_day_listen_pred  =  model_training_rf(df_train[day_lis],df_sub[day_lis])


     #切分資料建模
     day_listen_bins_model,  day_listen_bins_val_score,\
     day_listen_bins_pred  =  model_training_rf(df_train[day_lis_bins],df_sub[day_lis_bins])
```

- 分別計算其logloss

```
print("Original  score  :",orginal_day_listen_val_score)
print("Bins  score  :",day_listen_bins_val_score)
```

```
Original score : 0.193524822558668 96
Bins score : 0.1937750209986342
```

原始的loss比較小的原因可能是因為將資料做切分的同時過濾了noise，但也同時刪掉了一些有價值的資訓，所以我們傾向選擇原始資料

Model Structure
XGBoost

➢ **模型建立**

```
model  =  xgb.XGBClassifier(learning_rate=0.08,   max_depth=4,n_estimators=300,\
                            subsample=0.5,   seed=2,missing=-1)
model.fit(X_train,   Y_train,eval_set=xgb_watchlist,eval_metric='logloss',
                     early_stopping_rounds=20,verbose=70)
```

- Learning rate：每次迭帶的步長
- Max_depth：為樹的最大深度
- n_estimators：最大迭帶次數
- subsample：控制對於每棵樹，隨機采樣的比例
- seed：控制每次隨機數據的結果
- missing：將數據中缺失的值已-1為默認值
- eval_metric：logloss，代表隊於二元對數的損失
- early_stopping_rounds：用來控制模型過度擬和
- Train(訓練集) : Test(測試集) = 8 : 2

Model Structure
XGBoost

➢ **完整模型**

```
def model_training_xgb(training_data,testing_data):
    # splits train and validation set
    X = training_data.drop(labels=['msno','is_churn'],axis=1)
    Y = training_data['is_churn']
    X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size=0.2,random_state = 2)
    # model
    xgb_watchlist = [(X_train, Y_train), (X_val, Y_val)]
    model = xgb.XGBClassifier(learning_rate=0.08, max_depth=4,n_estimators=300,\
                              subsample=0.5, seed=2,missing=-1)
    model.fit(X_train, Y_train,eval_set=xgb_watchlist,eval_metric='logloss',
                       early_stopping_rounds=20,verbose=70)
    # caculating E_val

    model_probs = model.predict_proba(X_val)
        # [:,1] to show the prob to is_churn = 1
    model_val_score = log_loss(Y_val,model_probs[:,1])

    # predict on testing set
    model_pred_testing_set = model.predict_proba(testing_data.drop(labels=['msno','is_churn'],axis=1))
    model_pred_testing_set = model_pred_testing_set[:,1] # take out the prob if is_churn = 1
    submission = pd.DataFrame({"msno": testing_data.msno})
    submission.insert(1,column='is_churn',value=model_pred_testing_set)

    return model, model_val_score, submission
```

Model Structure
Summary

- **Random Forest和XGBoost比較**

```
rf_model,  rf_val_score,\
rf_pred  =  model_training_rf(train_df[parameters],test_df[parameters])

xgb_model,  xgb_val_score,\
xgb_pred  =  model_training_xgb(train_df[parameters],test_df[parameters])

#print  log_loss
print("log_loss  of  Random  Forest  :",rf_val_score)
print("log_loss  of  XGBoost  :",xgb_val_score)
```

這邊不做切分，因為從上一個看出使用原始資料的效果比較好

```
[0]    validation_0-logloss:0.629574    validation_1-logloss:0.629556
Multiple eval metrics have been passed: 'validation_1-logloss' will be used for early stopping.

Will train until validation_1-logloss hasn't improved in 20 rounds.
[70]    validation_0-logloss:0.165156    validation_1-logloss:0.165436
Stopping. Best iteration:
[95]    validation_0-logloss:0.164936    validation_1-logloss:0.165339

log_loss of Random Forest : 0.16848600995079138
log_loss of XGBoost : 0.16535620513393134
```

看出XGBOOST的log_loss較Randomforest還要小，因此我們認為XGBOOST比較好

➢ 模型比較

模型	優點	缺點
Random forest	1.訓練可以並行化，對於大規模樣本的訓練具有速度的優勢 2.由於進行隨機選擇劃分特徵列表，這樣在樣本維度較高的時候，仍然具有比較高的訓練效能 3. 由於存在隨機抽樣，訓練出來的模型方差小，泛化能力強 4. 對於部分特徵的缺失不敏感	1.每個節點要選擇特徵數量和決策樹的數量，所以更難裝配 2.在某些噪音比較大的特徵上容易陷入過擬合 3.取值比較多的劃分特徵對決策會產生更大的影響，從而可能影響模型的效果
XG Boost	由於通過優化目標函數導出了增強樹，基本上可以用來解決幾乎所有可以寫出漸變的目標函數	1.如果數據有noise，對過度擬合更敏感 2.由於樹木是按順序建造的，因此training通常需要更長時間

04

ANALYSIS

Training &Improvement

➢ Random Forest -1超參數調整及其優化結果

超參數	n_estimators	Min_sample_spl	結果 Log_loss	決策
entropy	**250**	**0.01**	**0.192628**	
entropy	**250**	**0.05**	**0.192535**	
entropy	**300**	**0.01**	**0.192628**	
entropy	**300**	**0.05**	**0.192535**	
gini	**250**	**0.01**	**0.192627**	
gini	**250**	**0.05**	**0.192534**	**v**
gini	**300**	**0.01**	**0.192627**	
gini	**300**	**0.05**	**0.192535**	

- n_estimators：最大迭帶次數
- min_samples_spl：內部節點再劃分所需最小樣本數

- XG Boost -1超參數調整及其優化結果

Learning rate	Max_depth	結果 Log_loss	決策
0.06	3	0.191613	
0.06	4	0.191613	
0.06	5	0.191613	
0.08	3	0.191609	
0.08	4	0.191609	
0.08	5	0.191609	
0.1	3	0.1916081	v
0.1	4	0.1916082	
0.1	5	0.1916082	

➢ Random Forest -2超參數調整及其優化結果

超參數	n_estimators	Min_sample_spl	結果 Log_loss	決策
entropy	**250**	**0.01**	**0.165946**	
entropy	**250**	**0.05**	**0.167968**	
entropy	**300**	**0.01**	**0.165948**	
entropy	**300**	**0.05**	**0.167949**	
gini	**250**	**0.01**	**0.165933**	**v**
gini	**250**	**0.05**	**0.167862**	
gini	**300**	**0.01**	**0.165939**	
gini	**300**	**0.05**	**0.167868**	

- n_estimators：最大迭帶次數
- min_samples_spl：內部節點再劃分所需最小樣本數

- XG Boost -2超參數調整及其優化結果

Learning rate	Max_depth	結果 Log_loss	決策
0.06	3	0.164585	
0.06	4	0.164588	
0.06	5	0.164600	
0.08	3	0.164593	
0.08	4	0.164582	v
0.08	5	0.164602	
0.1	3	0.164615	
0.1	4	0.164595	
0.1	5	0.164614	

➢ Data improvement

	First attempt	Second attempt	Improvement
Random Forest	**0.192534**	**0.165933**	**-0.026601**
XG Boost	**0.1916081**	**0.164582**	**-0.0270261**

- First attempt中的參數使用Day_listen and satisfication
- Second attempt在先前兩個參數的基準下，再加入註冊管道

Model	XG-BOOST-2
Learning Rate	0.1
Max_depth	3
Log_loss	0.164615

Model	XG-BOOST-2
Learning Rate	0.1
Max_depth	4
Log_loss	0.164595

Model	XG-BOOST-2
Learning Rate	0.08
Max_depth	4
Log_loss	0.164595

Model	Log_loss
Original	0.164615
Improve	0.164595

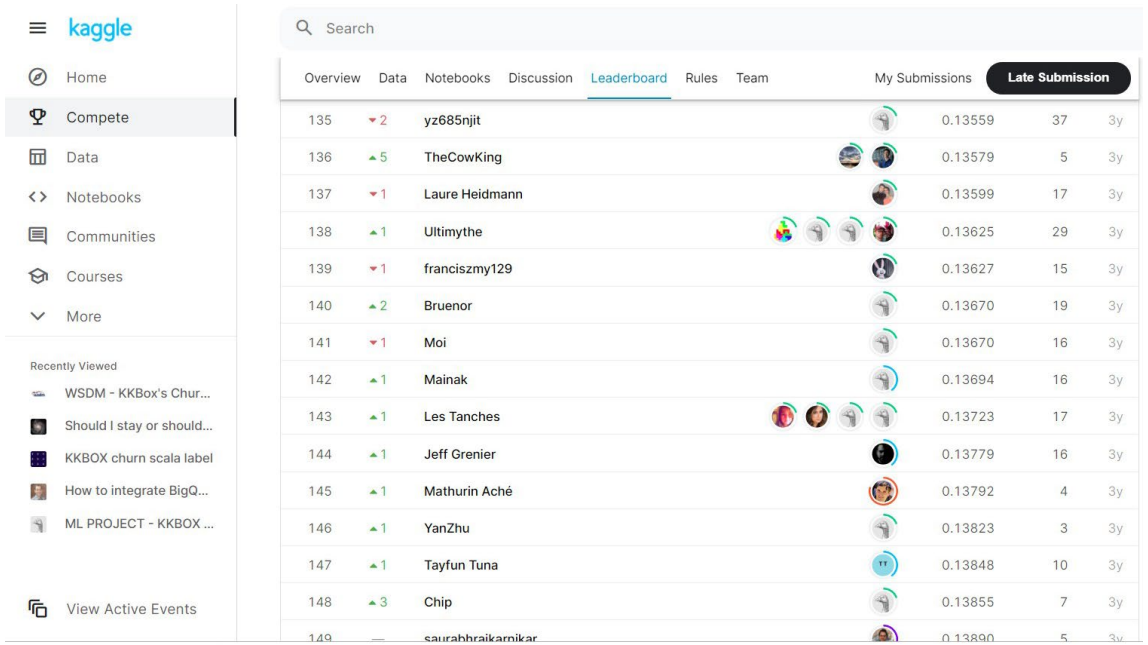
05 CONCLUSION

Conclusion
Kaggle Score

- **Data improvement**

	Kaggle Score
Random Forest	**0.13770**
XG Boost	**0.13184**

- 在574隊裡，取得約前20%的成績



Conclusion

Kaggle Score

➢未來展望

- ✓提供線上平台行銷策略建議
- ✓ 持續減少誤差，增加準確度
- ✓ 更多線上平台可應用

Thank you