作業六競賽

1. 小組姓名系級學號:

H24081163 統計 112 陳詠翰　　H24086032 統計 112 杜霙笙
D54099012 經濟 112 戰嬓

2. 競賽敘述和目標:

透過機器學習的方式根據銀行客戶的資料來預測客戶是否會流失(二元分類問題)

3. 資料前處理:

檢視是否有缺失值: 透過 train.info()的方式來看是否有缺失值與型態 而這次的資料沒有缺失值。
把 Gender 改成 binary 的形式，把 Geography 做 one-hot encoding。

4. 特徵處理與分析:

總流失率: 0.204
類別資料:

- **Country:** 來自德國的顧客的流失率是來自西班牙或法國的兩倍，德國的流失率大約為 32% 其他兩個國家大約 16% 17%
- **Gender:** 男生的流失率 16.6%，女生的流失率 25%。女生流失率比男生高。
- **Active member:** 是 active member 的流失率為 14% 而不是的流失率為 27%。
- **Number of products:** 可以注意的是 3 跟 4 的資料很少很少，但是當 numberofproducts 等於 3 時，流失率為 83%，等於 4 時流失率 100%，反而是等於 2 時的流失率不到 10%，等於 1 時的流失率為 28%。
- **Has credit card:** 這個有沒有信用卡的流失率似乎有流失率沒什麼直接相關，都是 20%左右。
- **Tenure:** 此變數最小是 2，最大是 7，而不動產的數量多寡似乎跟流失率沒有直接相關，大多數的顧客擁有 3-6 個不動產。

連續資料:

特徵處理:

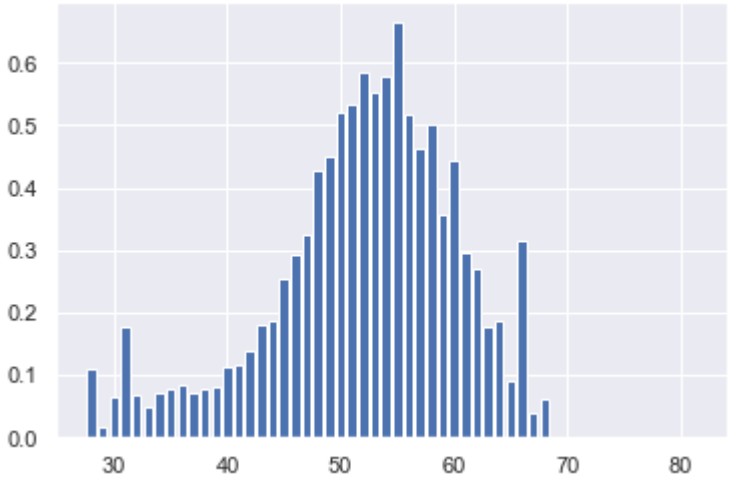
方法一: 把連續資料標準化

方法二: 除了 balance 以外，把連續資料透過手動切割。(依四分位距)

method 2 : cut the bins manually

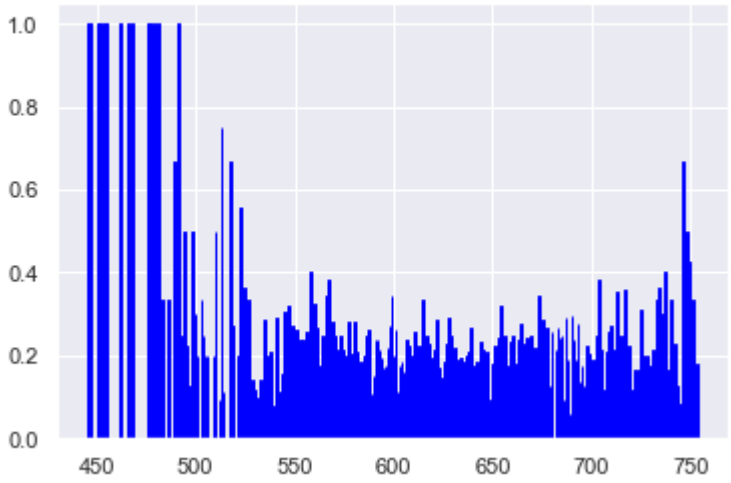
In [94]:
```
bins_age = [0, 38, 42, 47,99]
bins_creditscore = [0,590,631,672,1000]
bins_salary = [0,79217,100056,121105,200000]
bins_balance = [62397,62398,200000]
mylabel = ['1', '2', '3', '4']
mylabel2 = ['0','1']
train['age level'] = pd.cut(train['Age'], bins_age, include_lowest=True, labels=mylabel)
train['creditscorelevel'] = pd.cut(train['CreditScore'], bins_creditscore, include_lowest=True, labels=mylabel)
train['salarylevel'] = pd.cut(train['EstimatedSalary'], bins_salary, include_lowest=True, labels=mylabel)
train['balancelevel'] = pd.cut(train['Balance'], bins_balance, include_lowest=True, labels=mylabel2)
train
```

- **Age:** 從分布來看，大多數人都集中在 35-50 歲左右。而從流失率來看，我們可以看到 40 歲以下的人的流失率都偏低(基本上都在 10-15%左右)，40-50 歲這個區間裡面，隨著歲數越接近 50 歲，流失率越來越高，50 歲時流失率達到 5 成。50-60 歲的人流失率是所有年齡層中最高的，60 歲以上基本上隨著年齡的數字越來越大流失率也越來越小。



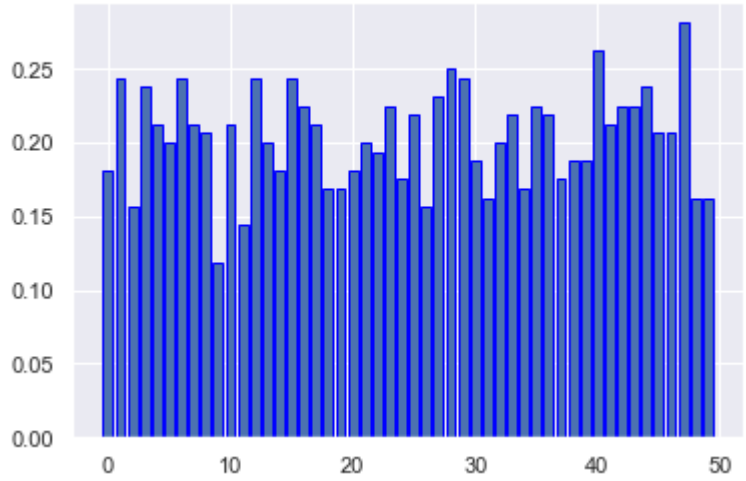
- **Balance:** 從分布來看，我們可以看到超過 30%的人的 balance 在 62397.41 這個值，這可能是代表 balance 的下限。也因此我們把 balance 切成等於 62397.41 跟不是這個值，而等於 62397.41 的流失率只有 13%，反而不是 62397.41 透過 qcut 切成五等分之後每個區間的流失率差異不大(大概在 20-25%)。
- **Credit Score:** 從分布來看，算是接近常態分配(有一點偏左)。從流失率來看，我們可以看到大約在 530 以下、740 以上的流失率比較高，尤其是 530 以下，基本上流失率接近 100%。

```
: <BarContainer object of 286 artists>
```



- **Estimated Salary:** 從流失率來看，我們切成 50 等分，我們可以看到不同等分之間的流失率有些微的差距，但也沒辦法明確說那些區間的流失率就一定比較小。

```
<BarContainer object of 50 artists>
```



另外， 我們也有利用 Pearson Correlation 判斷每個 column 的相關性，但效果不明顯，判斷不出相關程度的高低。

5. 預測訓練模型：

Feature 的調整（以 Random Forest 做為參考模型）:

模型	變數	結果
Random Forest	country 的 one-hot(3 個), isactivenumber, numberofproducts, gender, age level, balancelevel,	Acc: 0.8575 Pre: 0.6552 F1: 0.5714

	creditscorelevel(所有 level 都手動切割的變數)	**Overall: 0.6947**
Random Forest	country 的 one-hot, isactivenumber, numberofproducts, gender, age, balance, creditscore,salary (連續變數經過標準化)	Acc: 0.8475 Pre: 0.6334 F1: 0.5666 **Overall: 0.6709**

為了公平比較不同特徵組合，我們需要找到每個特徵組合最好的參數，因此我們使用 GridSearchCV 來做測試，程式碼如下:

All Features:

All Features

In [28]:
```
from sklearn.model_selection import GridSearchCV
x_train = train_data[['Gender','Tenure','NumOfProducts','HasCrCard',
       'IsActiveMember', 'Geography_France', 'Geography_Germany',
       'Geography_Spain', 'age level', 'creditscorelevel', 'salarylevel',
       'balancelevel']].values
y_train = train_labels.values
y_test = test_labels.values
x_test = test_data[['Gender','Tenure','NumOfProducts','HasCrCard',
       'IsActiveMember', 'Geography_France', 'Geography_Germany',
       'Geography_Spain', 'age level', 'creditscorelevel', 'salarylevel',
       'balancelevel']].values
from sklearn.ensemble import RandomForestClassifier
param_grid = {'max_depth': [3, 5, 6, 7, 8], 'max_features': [2,4,6,7,8,9],'n_estimators':[50,100],'min_samples_split': [3, 5, 6,
RanFor_grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5, refit=True, verbose=0)
RanFor_grid.fit(x_train,y_train)
print(RanFor_grid.best_params_)
```

```
{'max_depth': 6, 'max_features': 7, 'min_samples_split': 7, 'n_estimators': 100}
```

Show Results

In [29]:
```
par = list(RanFor_grid.best_params_.values())
model = RandomForestClassifier(max_depth = par[0],max_features = par[1],min_samples_split = par[2],n_estimators = par[3])
model.fit(x_train,y_train)
from sklearn.metrics import classification_report
print(classification_report(y_test, model.predict(x_test)))
```

```
              precision    recall  f1-score   support

           0       0.87      0.98      0.92      1594
           1       0.81      0.41      0.55       406

    accuracy                           0.86      2000
   macro avg       0.84      0.69      0.73      2000
weighted avg       0.86      0.86      0.84      2000
```

In [28]:
```
from sklearn.model_selection import GridSearchCV
x_train = train_data[['Gender','Tenure','NumOfProducts','HasCrCard',
       'IsActiveMember', 'Geography_France', 'Geography_Germany',
       'Geography_Spain', 'age level', 'creditscorelevel', 'salarylevel',
       'balancelevel']].values
y_train = train_labels.values
y_test = test_labels.values
x_test = test_data[['Gender','Tenure','NumOfProducts','HasCrCard',
       'IsActiveMember', 'Geography_France', 'Geography_Germany',
       'Geography_Spain', 'age level', 'creditscorelevel', 'salarylevel',
       'balancelevel']].values
from sklearn.ensemble import RandomForestClassifier
param_grid = {'max_depth': [3, 5, 6, 7, 8], 'max_features': [2,4,6,7,8,9],'n_estimators':[50,100],'min_samples_split': [3, 5, 6,
RanFor_grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5, refit=True, verbose=0)
RanFor_grid.fit(x_train,y_train)
print(RanFor_grid.best_params_)
```

```
{'max_depth': 6, 'max_features': 7, 'min_samples_split': 7, 'n_estimators': 100}
```

Drop Tenure 與 HasCreditCard:

drop Tenure

drop has creditcard

In [30]:
```
x_train = train_data[['Gender','NumOfProducts',
       'IsActiveMember', 'Geography_France', 'Geography_Germany',
       'Geography_Spain', 'age level', 'creditscorelevel', 'salarylevel',
       'balancelevel']].values
y_train = train_labels.values
y_test = test_labels.values
x_test = test_data[['Gender','NumOfProducts',
       'IsActiveMember', 'Geography_France', 'Geography_Germany',
       'Geography_Spain', 'age level', 'creditscorelevel', 'salarylevel',
       'balancelevel']].values
from sklearn.ensemble import RandomForestClassifier
param_grid = {'max_depth': [3, 5, 6, 7, 8], 'max_features': [2,4,6,7,8,9],'n_estimators':[50,100],'min_samples_split': [3, 5, 6,
RanFor_grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5, refit=True, verbose=0)
RanFor_grid.fit(x_train,y_train)
print(RanFor_grid.best_params_)
```

```
{'max_depth': 7, 'max_features': 4, 'min_samples_split': 3, 'n_estimators': 100}
```

Show Results

In [31]:
```
par = list(RanFor_grid.best_params_.values())
model = RandomForestClassifier(max_depth = par[0],max_features = par[1],min_samples_split = par[2],n_estimators = par[3])
model.fit(x_train,y_train)
from sklearn.metrics import classification_report
print(classification_report(y_test, model.predict(x_test)))
```

```
              precision    recall  f1-score   support

           0       0.86      0.98      0.92      1594
           1       0.82      0.39      0.53       406

    accuracy                           0.86      2000
   macro avg       0.84      0.68      0.72      2000
weighted avg       0.85      0.86      0.84      2000
```

經過不斷的嘗試後得到：

features : 'Gender','Tenure','NumOfProducts','HasCrCard',

'IsActiveMember', 'Geography_France', 'Geography_Germany', ¶

'Geography_Spain', 'scaleage', 'scalecredit', 'scalesalary',

'scalebalance'

In [32]:
```
from sklearn.model_selection import GridSearchCV
x_train = train_data[['Gender','Tenure','NumOfProducts','HasCrCard',
       'IsActiveMember', 'Geography_France', 'Geography_Germany',
       'Geography_Spain', 'scaleage', 'scalecredit', 'scalesalary',
       'scalebalance']].values
y_train = train_labels.values
y_test = test_labels.values
x_test = test_data[['Gender','Tenure','NumOfProducts','HasCrCard',
       'IsActiveMember', 'Geography_France', 'Geography_Germany',
       'Geography_Spain', 'scaleage', 'scalecredit', 'scalesalary',
       'scalebalance']].values
from sklearn.ensemble import RandomForestClassifier
param_grid = {'max_depth': [3, 5, 6, 7, 8], 'max_features': [2,4,6,7,8,9],'n_estimators':[50,100],'min_samples_split': [3, 5, 6,
RanFor_grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5, refit=True, verbose=0)
RanFor_grid.fit(x_train,y_train)
print(RanFor_grid.best_params_)
```

```
{'max_depth': 6, 'max_features': 6, 'min_samples_split': 7, 'n_estimators': 100}
```

Show Results

In [33]:
```
par = list(RanFor_grid.best_params_.values())
model3 = RandomForestClassifier(max_depth = par[0],max_features = par[1],min_samples_split = par[2],n_estimators = par[3])
model3.fit(x_train,y_train)
from sklearn.metrics import classification_report
print(classification_report(y_test, model3.predict(x_test)))
```

```
              precision    recall  f1-score   support

           0       0.87      0.97      0.92      1594
           1       0.81      0.44      0.57       406

    accuracy                           0.87      2000
   macro avg       0.84      0.71      0.75      2000
weighted avg       0.86      0.87      0.85      2000
```

不同模型與參數的測試（Drop 掉‘RowNumber’, ‘CustomerId’, ‘Surname’）：

分類模型	調整過的參數
Random Forest	n_estimators , max_features
Extra Trees	n_estimators, max_depth, min_samples_split
XGBoost	max_depth, eta, subsample
AdaBoost	n_estimators

6. 結果分析:

針對不同特徵處理方式:

我發現使用隨機森林，無論連續變數是用標準化的還是手動切割的效果都不是特別的好，把一些我認為沒影響的變數像是 hascreditcard 跟 tenure 加入模型之後效果也差不多。很有可能是因為老師上課說的 imbalanced data 的關係，或是沒有特別去調整隨機森林的參數。另外手動切割有個缺點就是萬一沒有切好的話可能會被分錯類別。

不同模型與參數的測試結果:

- **Random Forest**

參數調整順序: n_estimators => max_features

n_estimators	**Accuracy**	**Precision**	**F score**	**Final**
20	0.837	0.4464	0.5469	**0.6038**
30	0.8418	0.4385	0.55	**0.6041**
100	0.8485	0.4495	0.5666	**0.616**
200	0.8460	0.4479	0.5618	**0.6129**
300	0.8453	9.4527	0.5633	**0.6147**

max_features	**Accuracy**	**Precision**	**F score**	**Final**
3	0.8464	0.4495	0.5632	**0.6241**
6	0.8391	0.4621	0.5586	**0.6138**
10	0.8401	0.4685	0.5636	**0.618**

- **Extra Trees**

參數調整順序: n_estimators => max_depth => min_samples_split

n_estimators	**Accuracy**	**Precision**	**F score**	**Final**
35	0.8825	0.7673	0.6523	**0.7558**

40	0.8825	0.75	0.6421	**0.7462**
60	0.8875	0.7321	0.626	**0.7333**

max_depth	**Accuracy**	**Precision**	**F score**	**Final**
None	0.8698	0.7222	0.6182	**0.7249**
5	0.8775	0.7321	0.5869	**0.7176**
10	0.8584	0.7433	0.5951	**0.7186**
13	0.8599	0.7545	0.5987	**0.7238**

min_samples_split	**Accuracy**	**Precision**	**F score**	**Final**
3	0.8818	0.7689	0.6069	**0.738**
6	0.8875	0.7586	0.6617	**0.7693**
8	0.8799	0.7695	0.6142	**0.7405**
9	0.8787	0.7712	0.6093	**0.7387**

- **XGBoost**

  參數調整順序: eta => max_depth => subsample

eta	**Accuracy**	**Precision**	**F score**	**Final**
0.01	0.8675	0.72	0.576	**0.7067**
0.05	0.8675	0.7115	0.5827	**0.7068**
0.2	0.8575	0.65	0.5778	**0.6834**

max_depth	**Accuracy**	**Precision**	**F score**	**Final**
8	0.8705	0.7617	0.5769	**0.7204**
10	0.8725	0.7609	0.5785	**0.7214**
12	0.8733	0.7582	0.574	**0.7191**

subsample	**Accuracy**	**Precision**	**F score**	**Final**
1	0.8825	0.7917	0.6179	**0.7494**

- **AdaBoost**

n_estimators	**Accuracy**	**Precision**	**F score**	**Final**

65	0.871	0.805	0.6219	**0.7516**
100	0.8705	0.8017	0.5997	**0.7415**
200	0.87	0.7984	0.5988	**0.74**
300	0.8695	0.7975	0.5966	**0.7387**

Performance: Extra Trees > AdaBoost > XGBoost > Random Forest

Best Performance: Extra Trees(n_estimators = 35, max_depth = None, min_samples_split = 6)

7. 感想與心得:

陳詠翰: 從競賽中學到最多的應該就是怎麼完完整整的把機器學習的流程做一遍，雖然效果真的不是很好。從中我也查了一些到底要怎麼把連續資料做特徵處理，像是 log 或是手動切割資料之類的。最困難的地方就是要找到適合的模型讓表現變得更好吧，像我自己就是不太會調一些比較進階模型的參數所以表現一直就上不去。我花了不少時間在分析每個變數的分布跟流失率的關係，但是分析完之後就不太知道要怎麼做適合的特徵處理或是找到好的模型去套。我想應該是我時間花的不夠多的關係，期許自己之後暑假在 kaggle 裡面多多自學，提升自己這方面的能力。

杜霖笙: 不知道是不是因為是競賽的關係，感覺以但多了比較就多了一份刺激、多了一點想要知道到底怎麼樣會比較好的好奇心。在過程當中，我不斷的從網路上查不同的方法、模型、資料處理方式，也不斷嘗試不同的參數。其中，最令人難過的是即便花了很大的努力，結果也不一定如預期的進步。有時候實際嘗試新的東西，能力進步了，但 Accuracy, F score 都沒變，Precision 反而還降低了… 然後，有時候反而只是單純的因為好奇而嘗試的參數調整結果效果變好許多，真的會傻眼 XD。但我想，從這次經驗我學到了，了解理論與原理方面的東西一定對分析、判斷與找方向有幫助，但還是要不停地嘗試不同的東西才會有辦法發現最好的結果。

戰嫩：在這次的競賽中，我除了試過老師在課堂中有提到的模型外，也有額外找其他模型去測試。我覺得最有挑戰的部分是調整參數和選擇哪個 column 去做 train 的資料。我也有嘗試比較進階的模型 XGBoost，但我套進去後效果沒有想像中的好，大概就是因為沒有掌握好怎麼調整參數，對於參數的意義也沒有足夠的了解，才會沒讓 accuracy, precision, 和 f score 有明顯的提升。這次的競賽主要的收穫就是，讓我更了解如何使用 scikit learn 這個套件，也養成了去看官方網站查 function 的習慣，還有下關鍵字解決問題的能力。