FinTech 2022

Final Project Report

組別：黑貓站在檸檬上看狗狗吃蛋糕
組員：　R11922003 許育騰
　　R09525109 陳顥
　　B09703096 徐雋翔
　　B08901158 吳詩昀

1. Missing Value

在原始資料中，我們發現有缺失值的欄位包含：

- occupation_code：類別
- tx_amt：數值
- fiscTxId：類別
- txbranch：類別

其中 tx_amt 為數值型，其餘皆為類別型，針對不同類型的資料分別嘗試過以下幾種方法來處理缺失值：

- 數值型：直接刪除、補中位數、根據 debit_credit 補對應的中位數、MICE
- 類別型：直接刪除、忽略 (one-hot encoding 時會被轉成零向量)、MICE

根據 validation set 和 public score 的結果，上述方法的優劣排序為：

- 數值型：MICE > 根據 debit_credit 補對應的中位數 > 補中位數 > 直接刪除
- 類別型：MICE > 忽略 (one-hot encoding 時會被轉成零向量) > 直接刪除

依據以上結果，我們最終選擇用 **MICE** 作為兩種類型資料的填補方式

MICE 補缺值的方式，是假設所有欄位為 V1, V2, V3 … Vn，當我們要對 V1 的欄位補缺值時，會將 V2, V3 … Vn 作為自變數 (x)，把 V1 當作應變數 (y)，進行建模來預測結果填補 V1 的缺值。

以下描述使用 public_train_x_dp_full_hashed.csv 檔作為輸入，MICE 補值的過程：

- 我們使用的是叫做 miceforest 的 package，首先讀 csv 檔並確認缺值數量。

```
dp = pd.read_csv('private_x_dp_full_hashed.csv')
print(dp.isnull().sum(0))
```

```
cust_id                0
debit_credit           0
tx_date                0
tx_time                0
tx_type                0
tx_amt               634
exchg_rate             0
info_asset_code        0
fiscTxId           19464
txbranch           18164
cross_bank             0
ATM                    0
dtype: int64
```

- 輸入的 dataframe 需要所有欄位為類別型或數值型，所以先做一些轉換處理，將 cust_id 轉為類別型，以及 debit_credit 轉為編號：CR=0，DB=1。

```
dp.cust_id = dp.cust_id.astype('category')
dp = dp.replace({'CR': 0, 'DB': 1})
```

- 接著就能送進MICE去做補值，並確認最後結果是否有完成補值。

```
# Create kernel.
kds = mf.ImputationKernel(dp, save_all_iterations=True)
# Run the MICE algorithm for 2 iterations
kds.mice(2)
# Return the completed dataset.
dp_complete = kds.complete_data()
print(dp_complete.isnull().sum(0))
```

```
debit_credit       0
tx_date            0
tx_time            0
tx_type            0
tx_amt             0
exchg_rate         0
info_asset_code    0
fiscTxId           0
txbranch           0
cross_bank         0
ATM                0
dtype: int64
```

可以比較一下使用 MICE 補值前後的差異。

補值前：

	A	B	C	D	E	F	G	H	I	J	K	L
1	cust_id	debit_cred	tx_date	tx_time	tx_type	tx_amt	exchg_rate	info_asset_	fiscTxId	txbranch	cross_bank	ATM
2	01720565	CR	36	18	2	68265	1	16			0	0
3	01720565	CR	42	17	2	932058	1	16			0	0
4	01720565	CR	39	18	2	6089	1	16			0	0
5	01720565	CR	49	15	2	776715	1	16			0	0
6	01720565	CR	70	19	2	61630	1	16			0	0
7	01720565	CR	64	17	2	4992	1	16			0	0
8	01720565	CR	56	14	2	6570	1	16			0	0
9	01720565	CR	64	17	2	14235	1	16			0	0
10	01720565	CR	264	0	2	25830	1	16			0	0
11	01720565	CR	68	15	2	103572	1	16			0	0
12	01720565	CR	70	19	2	20712	1	16			0	0
13	01720565	DB	75	1	2	1553430	1	16			0	0
14	01720565	DB	42	2	2	93206	1	16			0	0
15	01720565	CR	39	17	2	31069	1	16			0	0
16	01720565	DB	12	2	2	10356	1	16			0	0
17	01720565	CR	7	18	2	207124	1	16			0	0
18	01720565	CR	11	16	2	29083	1	16			0	0
19	01720565	CR	7	18	2	1470580	1	16			0	0
20	01720565	CR	64	17	2	103572	1	16			0	0
21	01720565	CR	21	14	2	3106860	1	16			0	0
22	01720565	CR	14	15	2	776715	1	16			0	0
23	01720565	CR	0	14	2	80789	1	16			0	0

補值後：

	A	B	C	D	E	F	G	H	I	J	K	L
1	cust_id	debit_cred	tx_date	tx_time	tx_type	tx_amt	exchg_rate	info_asset_	fiscTxId	txbranch	cross_bank	ATM
2	01720565	0	36	18	2	68265	1	16	22	0	0	0
3	01720565	0	42	17	2	932058	1	16	22	68	0	0
4	01720565	0	39	18	2	6089	1	16	22	0	0	0
5	01720565	0	49	15	2	776715	1	16	22	68	0	0
6	01720565	0	70	19	2	61630	1	16	22	0	0	0
7	01720565	0	64	17	2	4992	1	16	5	0	0	0
8	01720565	0	56	14	2	6570	1	16	20	45	0	0
9	01720565	0	64	17	2	14235	1	16	22	0	0	0
10	01720565	0	264	0	2	25830	1	16	22	80	0	0
11	01720565	0	68	15	2	103572	1	16	22	71	0	0
12	01720565	0	70	19	2	20712	1	16	22	12	0	0
13	01720565	1	75	1	2	1553430	1	16	4	0	0	0
14	01720565	1	42	2	2	93206	1	16	4	0	0	0
15	01720565	0	39	17	2	31069	1	16	22	0	0	0
16	01720565	1	12	2	2	10356	1	16	4	0	0	0
17	01720565	0	7	18	2	207124	1	16	22	0	0	0
18	01720565	0	11	16	2	29083	1	16	22	0	0	0
19	01720565	0	7	18	2	1470580	1	16	5	0	0	0
20	01720565	0	64	17	2	103572	1	16	22	82	0	0
21	01720565	0	21	14	2	3106860	1	16	22	126	0	0
22	01720565	0	14	15	2	776715	1	16	5	0	0	0
23	01720565	0	0	14	2	80789	1	16	22	0	0	0

2. Feature Selection and Preprocessing

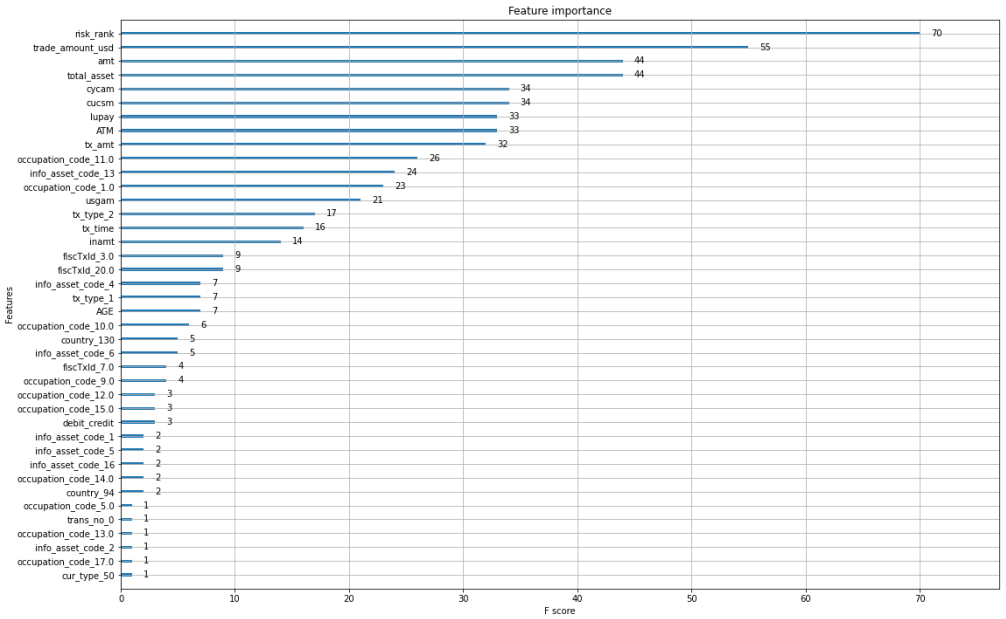
1) Feature Selection

- 人工觀察 SAR 為 1 的情況：

  將 alert_date 前的交易筆數相加起來可以發現，label == 1 的在 train 資料集當中，將其數值型資料分箱，與 label == 0 相比，可以觀察到有較大的趨勢。雖然在 public 公布答案後，趨勢有稍微不明顯一點，但仍舊可以往這方面來做嘗試。

  於是就決定試著將大部分數值型的資料進行手動分箱觀察看看，並試著預測看看結果。得出如果是將全部資料都分箱改成類別型的話，結果雖然相較先前好，但頂多就只有 public score 進步 0.001 而已，可是如果只針對比較重要的 feature 如：tx_amt, trade_amount_usd 等 feature 下去處理，結果可以好大概 0.004

- 使用 XGBClassifier 進行初步訓練，並計算 feature importance，結果如下



根據以上結果，刪除相對不重要的 **features**，包含 **clamt, csamt, cucah, country, cur_type, exchg_rate, fiscTxId, txbranch, trans_no**，並使用剩餘的來做後續訓練

2) Feature Preprocessing

針對類別型資料又再細分為有序和無序兩種類型:

- 有序型:risk_rank, AGE, debit_credit, tx_time, cross_bank, ATM
- 無序型:country, cur_type, occupation_code, tx_type, info_asset_code, fiscTxId, trans_no

有序型直接當作數值型，無序型則是會做 one-hot encoding，接著再把所有資料透過以下三種方法來處理:

- 全部標準化
- 全部由 FAMD (Factor Analysis of Mixed Data) 壓縮成較低維度的向量
- 只有 one-hot encoding 的結果會經過 FAMD 降維，其餘則是標準化

根據 validation set 和 public score 的結果，最後一種「只有 **one-hot encoding** 的結果會經過 **FAMD** 降維，其餘則是標準化」的方法效果最好，並最為我們最終的處理方法。

3. Machine Learning Methodology

將整個問題視為二元分類的任務，並把 SAR 為 1 類別的機率當作最後提交的答案，模型的部分嘗試過以下三種：

1) XGBClassifier

首先用 alert_key 至 custinfo 中找到對應的 cust_id，接著再到 ccba, cdtx, dp, remit 中找出其他資料 (最接近發生日期的那一筆)，並把這些資料串在一起後輸入給 XGBClassifier，參數皆使用預設值

2) DNN (Linear)

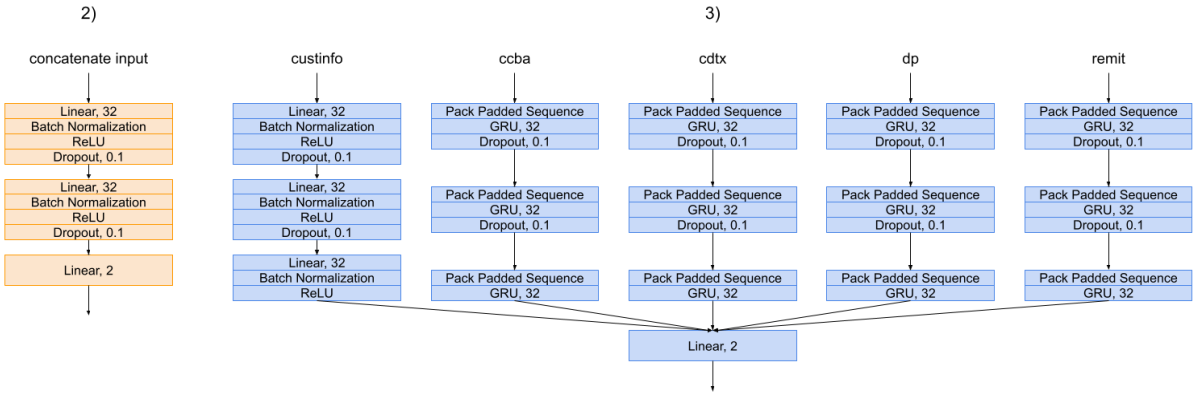
首先用 alert_key 至 custinfo 中找到對應的 cust_id，接著再到 ccba, cdtx, dp, remit 中找出其他資料 (最接近發生日期的那一筆)，並把這些資料串在一起後輸入給 Linear

3) DNN (Linear + GRU)

首先用 alert_key 至 custinfo 中找到對應的 cust_id，接著再到 ccba, cdtx, dp, remit 中找出其他資料 (發生日期前 30 天的所有資料，並取最後 64 筆當作時序資料)，之後將 custinfo 的資料輸入 Linear，其餘時序資料各自輸入給 GRU，再將各個子模型的輸出串在一起，並輸入另一個 Linear

其中 2) 和 3) 共用以下設定，而模型結構細節如下圖

- 64 batch size
- 1e-3 learning rate
- AdamW optimizer (weight_decay=1e-2)
- Cross Entropy Loss
- Step decay learning rate scheduler (gamma=0.95)
- 20 epochs early stop on precision of recall@N-1 score



根據 validation set 和 public score 的結果，**DNN (Linear + GRU)** 的效果最好

4. Others

針對 imbalanced data 的問題，嘗試過以下幾種方法

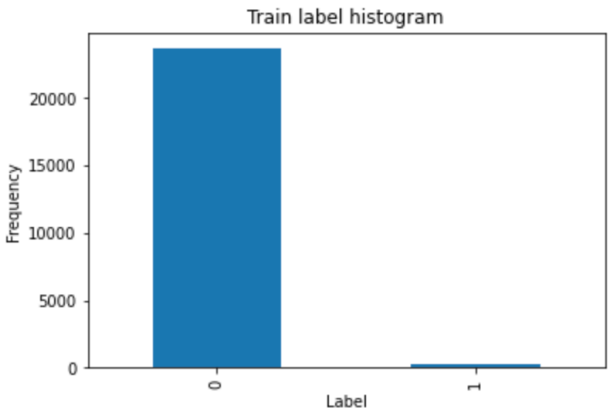
1) Different Loss Function

- **Weighted Cross Entropy Loss**：在原本的 Cross Entropy Loss 中乘上各類別的權重，公式為 $L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} w_j y_{ij} log(p_{ij})$，其中 N 為資料數量，C 為類別數量，w_j 為 j 類別的權重，y_{ij} 只有當資料 i 屬於 j 類別時為 1，其餘則為 0，p_{ij} 為資料 i 屬於 j 類別的預測機率
- **F1-Score Loss**：若資料屬於 SAR 為 1 類別的預測機率是 0.6，則將其視為 0.6 個 positive 和 0.4 個 negative，並利用此方式計算 F1-Score，再將 1 扣掉 F1-Score 當作 Loss
- **Focal Loss**：公式為 $L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} w_j y_{ij}(1 - p_{ij})^{\gamma} log(p_{ij})$，其中 N 為資料數量，C 為類別數量，w_j 為 j 類別的權重，y_{ij} 只有當資料 i 屬於 j 類別時為 1，其餘則為 0，γ 為可調參數，數值越大越能降低易分類資料的權重 ，p_{ij} 為資料 i 屬於 j 類別的預測機率

根據 validation set 和 public score 的結果，使用 **Focal Loss** 的效果最好

2) SMOTE

將 data 進行繪圖後，發現 label 極為不平均，如下圖



處理不平衡數據的一種方法是在訓練之前通過丟棄多數類來平衡數據，也就是 under-sampling。但是欠採樣的缺點是，以這種方式訓練的模型在真實世界的偏斜測試數據上表現不佳，因為幾乎所有信息都被丟棄了。

更好的方法是對少數類別進行 over-sampling。因此我們嘗試將 training dataset 以 SMOTE (Synthetic Minority Over-sampling Technique) 對少數類的樣本進行分析，並合成新樣本添加到數據集中再進行訓練。

- 我們使用了 imblearn 套件中的 SMOTE

```
from imblearn.over_sampling import SMOTE
```

- 接著在產生完 train_data 及 vali_data 後對 train_data 使用 SMOTE

```
train_data, vali_data, train_label, vali_label = train_test_split(all_data, a
train_data, train_label = apply_SMOTE(train_data, train_label)
```

```
def apply_SMOTE(X_train, y_train):
  print("Applying SMOTE...\n")
  print("Before OverSampling, counts of label '1': {}".format(y_train.count(torch.tensor(1))))
  print("Before OverSampling, counts of label '0': {} \n".format(y_train.count(torch.tensor(0))))

  sm = SMOTE(random_state=2)
  X_train_res, y_train_res = sm.fit_resample(torch.stack(X_train, 0), y_train)

  print('After OverSampling, the shape of train_X: {} {}'.format(len(X_train_res), X_train_res.shape))
  print('After OverSampling, the shape of train_y: {}\n'.format(len(y_train_res)))

  print("After OverSampling, counts of label '1': {}".format(y_train_res.count(torch.tensor(1))))
  print("After OverSampling, counts of label '0': {}\n".format(y_train_res.count(torch.tensor(0))))
  return X_train_res, y_train_res
```

- 在 over-sampaling 前後的 train_data 比較如下：

```
Applying SMOTE...

Before OverSampling, counts of label '1': 187
Before OverSampling, counts of label '0': 18937

After OverSampling, the shape of train_X: 37874 (37874, 53)
After OverSampling, the shape of train_y: 37874

After OverSampling, counts of label '1': 18937
After OverSampling, counts of label '0': 18937
```

但經過反覆測試，validation set 和 public score 的成績皆不如以原始數據訓練來的理想，所以並未採用

5. Conclusion

我們最終提交出去的結果是以以下方式產出：

- 針對有缺失值的欄位，包含 occupation_code, tx_amt, fiscTxId, txbranch，用 MICE 作為缺失資料的填補方式
- 使用 XGBClassifier 進行訓練並計算 feature importance，刪除相對不重要的 features，包含 clamt, csamt, cucah, country, cur_type, exchg_rate, fiscTxId, txbranch, trans_no
- 將 one-hot encoding 的結果經過 FAMD 降維，其餘資料則是標準化
- 利用 DNN (Linear + GRU) 模型及 Focal Loss 進行訓練，並產出最終結果

Contribution of Team Members

許育騰	*Report, Feature Preprocessing, Machine Learning Methodology, Different Loss Function*
陳顥	*Report, Missing Value*
徐雋翔	*Report, Feature Selection*
吳詩昀	*Report, SMOTE*